Universidade Estadual de Campinas
Instituto de Computação

Felipe dos Santos Pinto de Andrade

Evocycle: Vida Artificial em um Ecossistema de Algoritmos Bio-inspirados

CAMPINAS
2021

Felipe dos Santos Pinto de Andrade

Evocycle: Vida Artificial em um Ecossistema de Algoritmos Bio-inspirados

Tese apresentada ao Instituto de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Ricardo da Silva Torres

Este exemplar corresponde à versão final da Tese defendida por Felipe dos Santos Pinto de Andrade e orientada pelo Prof. Dr. Ricardo da Silva Torres.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

An24e

Andrade, Felipe dos Santos Pinto de, 1986-
Evocycle : vida artificial em um ecossistema de algoritmos bio-inspirados / Felipe dos Santos Pinto de Andrade. – Campinas, SP : [s.n.], 2021.

Orientador: Ricardo da Silva Torres.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Computação.

1. Computação natural. 2. Vida artificial. 3. Computação evolutiva. 4. Ecossistema. I. Torres, Ricardo da Silva, 1977-. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Evocycle : artificial life in an ecosystem of bio-inspired algorithms
Palavras-chave em inglês:
Natural computation
Artificial life
Evolutionary computation
Biotic communities
Área de concentração: Ciência da Computação
Titulação: Doutor em Ciência da Computação
Banca examinadora:
Ricardo da Silva Torres [Orientador]
Rafael Stuns Parpinelli
Douglas Rodrigues
Ulisses Martins Dias
Guilherme Palermo Coelho
Data de defesa: 30-07-2021
Programa de Pós-Graduação: Ciência da Computação

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-1536-1418
- Currículo Lattes do autor: http://lattes.cnpq.br/3183352170573277

Universidade Estadual de Campinas
Instituto de Computação

Felipe dos Santos Pinto de Andrade

Evocycle: Vida Artificial em um Ecossistema de Algoritmos Bio-inspirados

Banca Examinadora:

- Prof. Dr. Ricardo da Silva Torres - Presidente
  IC/UNICAMP
- Prof. Dr. Rafael Stubs Parpinelli
  CCT/UDESC
- Dr. Douglas Rodrigues
  FC/UNESP
- Prof. Dr. Ulisses Martins Dias
  FT/UNICAMP
- Prof. Dr. Guilherme Palermo Coelho
  FT/UNICAMP

A ata da defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

Campinas, 30 de julho de 2021

Look! It's moving. It's alive.
It's alive... It's alive, it's moving,
it's alive, it's alive, it's alive, it's alive,
IT'S ALIVE!
(Frankenstein, Direção: James Whale, 1931)

Agradecimentos

O período de doutoramento foi uma jornada que não seria possível de ser finalizada sem a contribuição de muitas pessoas que atravessaram a minha vida. Por este motivo, devo sinceros agradecimentos a muitas dessas pessoas.

Em primeiro lugar, gostaria de agradecer meu orientador, Ricardo da Silva Torres, que acreditou em mim desde começo quando começamos a trabalhar juntos com uma iniciação científica, ainda na minha graduação. Se hoje esse doutorado está completo, é por ter acreditado em mim. Obrigado!

Agradeço também ao professor Claus Aranha, da universidade de Tsukuba, onde passei um período, o chamado "doutorado sanduíche". Além da grande contribuição na pesquisa, também foi a oportunidade de realizar um sonho de conhecer o Japão. Por toda a ajuda, obrigado!

Agradeço também a minha família, meus pais Carmén e Adauto, e minha irmã Vanessa, meu irmão Kauê e todos os outros membros da família, pelo apoio e estrutura na vida. Obrigado!

Também mando meus agradecimentos a todos os amigos, Rômulo, Bruno, Bosso, Kívia, Miki, Buiu, Arthur, Victor, Renato, Daniel, Maíra, Eduardo, Bea, Marie, Amaury, Rebeca, Nat, Nath, Aldenir, Noel, Paulo, Henrique, Tadashi, Fer, Bibs, Caro, Luis, Tom, Dani, Decker, Diduch, Rafa, João, Karina, Carol, Mark, Nê, Rodrigo, e todas as pessoas que integram meus afetos que porventura não estão listados, pois sem elas não vale a pena fazer essa caminhada. Obrigado!

Em especial agradeço também ao pessoal da secretaria de pós graduação do IC, por serem sempre solícitos e eficientes em tudo que precisei. Muito Obrigado!

Por fim, agradeço as agências de fomento que viabilizaram a pesquisa, pois o presente trabalho foi parcialmente financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e processo 88881.135859/2016-01. Este trabalho também foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico no processo 141224/2016-9. Parte deste trabalho também contou com apoio da Norwegian University of Science and Technology (NTNU).

Resumo

Vida artificial é a área de estudo que tem por objetivo reproduzir comportamentos análogos à vida em diferentes tipos de mídia. Na abordagem *soft*, essa tarefa é realizada usando simulações em *software*. Essa tese apresenta um arcabouço de vida artificial, chamado Evocycle, que utiliza diferentes meta-heurísticas bio-inspiradas como agentes do sistema. Tais meta-heurísticas são parte do campo da Computação Evolutiva em Ciência da Computação, na qual a teoria biológica da evolução é tomada como inspiração para projetar algoritmos estocásticos de otimização. Na computação evolutiva clássica, frequentemente abordagens elitistas são utilizadas, como por exemplo usar funções de *fitness* para selecionar indivíduos para novas gerações. No entanto, a seleção natural como observada na natureza não possui uma função de *fitness* para avaliação.

Neste trabalho, propomos um arcabouço que remove a ênfase do processo de seleção baseado apenas nos valores de *fitness*, e tenta uma abordagem diferente, baseada em relações ecológicas de alimentação, reprodução e predação, utilizando assim o valor de *fitness* indiretamente. Nosso sistema de vida artificial forma um ecossistema no qual diferentes espécies são modeladas pela hibridização de algoritmos de inteligência coletiva e evolucionários, e a interação dessas espécies através das relações ecológicas constrói seu comportamento ao longo do tempo. Com essa abordagem, buscamos aumentar a plausibilidade biológica do sistema evolutivo em nossas simulações de vida artificial.

Para validar o arcabouço proposto, nós o simulamos com implementações utilizando os algoritmos de *Particle Swarm Optimization* (PSO), no qual a função de velocidade de cada partícula é definida por uma árvore de programação genética (GP), e o algoritmo de otimização *Artificial Bee Colony* (ABC). Ao invés de um ciclo completo de seleção por função de *fitness*, as árvores GP, que codificam cada partícula PSO, evoluem tomando como base a sobrevivência dentro da simulação, levando em conta fatores como energia interna, idade e proximidade de outros indivíduos.

Este ecossistema foi estudado em *benchmarks* de otimização contínua, e pela análise da variação da frequência genética de populações de interesse, nós observamos o desenvolvimento de dinâmicas ecológicas promissoras, resultado da pressão seletiva oriunda das interações ecológicas entre os agentes do sistema de vida artificial. Também avaliamos a resiliência da população de predadores no sistema utilizando estudos da teoria de catástrofes em Biologia. As principais contribuições deste trabalho consistem na proposta do sistema de vida artificial em si, o estudo da emergência do comportamento de busca dos agentes do sistema em um cenário evolutivo que buscava maior plausibilidade biológica, e a proposta de uso da análise das taxas de recuperação após perturbações na população para avaliar sua resiliência.

Abstract

Artificial life is the field of study that aims to reproduce life-like behavior in different types of media. In the soft approach, this task is performed using software simulations. This thesis presents an artificial life framework, named EvoCycle, which uses different bio-inspired meta-heuristics as agents of the system. Such meta-heuristics are part of the Evolutionary Computation field in Computer Science, in which the general theory of evolution is taken as inspiration to design stochastic optimization algorithms. In classic evolutionary computation, often elitist approaches are used, such as those based on fitness functions, to select individuals for new generations. However, natural selection as observed has no explicit fitness evaluation.

In this work, we propose a framework that de-emphasizes the selection process based solely on fitness values, and tries a different approach based on ecological relationships of feeding, reproduction, and predation, using the fitness values indirectly. Our artificial life system composes an ecosystem where different species are modelled as a hybridization of swarm intelligence with evolutionary algorithms, and the interaction of the species through ecological relationships shapes their behaviour over time. With this approach, we seek to increase the biological plausibility of the evolution process in our artificial life simulations.

To validate the proposed framework, we performed simulations with implementations based on the Particle Swarm Optimization (PSO) algorithm, in which the velocity rule of each particle is defined by a Genetic Programming (GP) tree and the Artificial Bee Colony (ABC) algorithm. Instead of an explicit fitness-selection cycle, the GP trees composing individual PSO particles evolve based on surviving in the simulation, taking into account different factors, such as energy, hunger, mutation, and proximity to other individuals.

This ecosystem is investigated on optimization benchmarks, and through the analysis of the variance of the gene frequency, we observed the development of promising ecological dynamics, resulting from the selective pressure coming from the ecological relationships between agents of the artificial life system. We also evaluate the resilience of a predator population in the system using biological studies of the catastrophe theory. The main contributions of the work are the proposal of the artificial life framework itself, the study of the emergence of search behavior of the agents in an evolutionary scenario that aimed to be more biologically plausible, and the proposal of using the analysis of recovery rates after perturbations to investigate the resilience of artificial populations.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Desde o seu princípio, a Ciência da Computação busca a reprodução da capacidade de se obter conhecimento de forma autônoma. Em seu trabalho pioneiro, por exemplo, Alan Turing propôs que máquinas poderiam ser inteligentes, através de um processo de aprendizado ou de evolução [92]. Alinhado a estes conceitos, o grande campo da Inteligência Artificial [77] se estabelece, lidando com desafios de pesquisa associados com a criação de programas de computadores com essa capacidade. Com os avanços computacionais, ocorreu uma fragmentação desse campo em diversos segmentos de acordo com objetivos específicos, como Aprendizado de Máquina [63], Vida Artificial [52], Computação Evolutiva [8], Visão Computacional [33], Robótica [65], Processamento de Linguagem Natural [16], Planejamento Automatizado [54], entre outros.

Muitos destes campos tomam como inspiração processos observados no mundo natural, podendo ser agrupados no campo conhecido como Computação Natural [22]. Nesta categoria encontramos três tipos de estratégias que relacionam computação e processos do mundo natural. Na primeira delas, processos do mundo natural servem de base de inspiração para a realização de computação, como nas Redes Neurais Artificiais (ANN), Computação Evolutiva (EC) e Sistemas Imunes Artificiais. Na segunda, a computação é utilizada para realizar simulações e emulações de processos naturais, como na Vida Artificial (ALife). Finalmente, na terceira, materiais naturais são utilizados para a realização de computação, ao invés de componentes eletrônicos. De uma forma geral, técnicas de Computação Natural que envolvem o processo evolutivo constituem o elemento que foi escolhido como objeto de estudos nesta tese.

Tomando como ponto de partida a Computação Evolutiva, temos o campo dominado por algoritmos projetados para resolver problemas e que tomam inspiração no processo de evolução. Diversos algoritmos foram desenvolvidos que, em geral, seguem as seguintes regras:

1. Criar uma população de soluções aleatoriamente.

2. Avaliar os indivíduos desta população conforme sua capacidade de resolver um problema em questão.

3. Selecionar os melhores indivíduos desta população, ou seja, os que resolvem o problema segundo a avaliação do passo anterior.

4. Aplicar operações nos indivíduos selecionados que possam, ou gerar novos indivíduos, diferentes dos que existiam previamente, ou alterar os indivíduos já existentes, tentando assim, se aproximar de uma solução melhor.

5. Gerar uma nova população com os indivíduos dos dois passos anteriores.

6. Repetir os últimos 4 passos até que se atinja um critério de parada pré-determinado.

Para que estes passos possam ser realizados, alguns elementos que compõem os algoritmos de EC precisam ser definidos:

- Representação: dado um determinado problema tratado por um algoritmo de EC, a representação é a maneira em que uma solução candidata para o problema é codificada em uma estrutura de dados. Assim, temos esta estrutura de dados funcionando como o genótipo (a codificação genética de um indivíduo) e o seu papel como solução para o problema o fenótipo (característica da expressão de um genótipo).

- *Fitness:* uma medida de quão bem a soulução candidata resolve o problema tratado. De maneira geral, é feito o uso de uma função que avalia cada um dos indivíduos da populaç ao de soluções candidatas.

Assim, a partir da avaliação de uma solução (fenótipo), alterações são realizadas em sua estrutura (genótipo). Os algoritmos evolutivos se baseiam portanto no mapeamento genótipo-fenótipo para tentar encontrar sempre melhores soluções do que as existentes previamente.

Em uma outra abordagem, temo o campo da Vida Artificial (do inglês, *Artificial Life* – AL), que compartilha suas origens com a da Computação Evolutiva (do inglês *Evolutionary Computation* – EC). Ambos tomam como inspiração processos que ocorrem no mundo natural para simular ou realizar procedimentos computacionais. Desde sua proposição, as duas áreas se desenvolveram consideravelmente com alguma sobreposição em diversos cenários.

Como definido em [53], AL busca a construção de sistemas que emulam as características de sistemas vivos naturais. Para isso, são utilizadas mídias artificiais para sintetizar comportamentos análogos à vida. Dessa forma, o estudo da AL contempla elementos que podem auxiliar a melhor compreensão dos sistemas de vida que conhecemos, como no campo da Biologia, e também na expansão do conhecimento para possíveis sistemas de vida, diferentes de como a conhecemos. Em resumo, AL se concentra em gerar comportamento análogo à vida, para obter conhecimento dos sistemas de vida e EC se concentra em resolver problemas da melhor forma possível (otimização).

Estes dois propósitos distintos geram um impacto no projeto e comportamento de sistemas de AL e EC. Como consequência do ideal de gerar comportamento análogo à vida, auto-organização é uma característica tipicamente encontrada em sistemas de AL.

O conceito de auto-organização se refere ao processo de emergência de um padrão, derivado de interações locais dos agentes pertencentes ao sistema. Tal padrão surge espontaneamente, de forma decentralizada e distribuída. Por essas razões, em uma categorização mais ampla, AL pode ser descrita como um sistema complexo adaptativo (do

inglês *Complex Adaptive Systems* – CAS) [13]. Em tais sistemas, as interações entre as populações de agentes adaptativos resultam em dinâmicas complexas, produzindo um fenômeno emergente. Quando essa emergência consiste de um comportamento coletivo de uma população, ele é denominado auto-organização. Neste caso, agentes adaptativos respondem a estímulos de outros agentes ou do ambiente[1].

Algoritmos de EC se concentram em resolver problemas usando como inspiração a teoria evolutiva, a partir do mapeamento do processo de descoberta de soluções em processos evolutivos. As abordagens clássicas são os Algoritmos Evolutivos (do inglês *Evolutionary Algorithms – EA*), que, de forma geral, não têm a propriedade de auto-organização. Ao invés disso, uma função de *fitness* externa exerce a pressão seletiva para selecionar as melhores soluções (indivíduos) dentro de uma população. Uma vez que a função de *fitness* é externa à população de agentes, não podemos considerar que estes sistemas possuem a característica de auto-organização [76].

Apesar disso, na área de Computação Natural, existem algoritmos que exploram a auto-organização na tarefa de resolução de problemas, como os algoritmos de inteligência coletiva (do inglês *Swarm Inteligence – SI*)[2] e Algoritmos Meméticos. Essa classe de algoritmos, no entanto, se organiza a partir de regras internas, simulando um comportamento social coletivo, geralmente inspirado por fenômenos biológicos.

Apesar da inspiração do processo de computação evolutiva ser baseado na evolução natural, existem diferenças entre o fenômeno observado e sua aplicação em EC. Tal distância entre o processo natural e os métodos inspirados é chamada de plausibilidade biológica [70]. Um dos pontos divergentes é relativo ao tratamento do *fitness*, medida utilizada para avaliar o quanto um indivíduo de uma população se encontra apto a diversas situações, assim como uma seleção de indivíduos diretamente relacionada a esse valor. Com este contexto, passamos a refletir acerca das implicações de modificar a maneira como *fitness* e seleção são empregados na EC e verificar os efeitos de aumentar a plausibilidade biológica do processo evolutivo como um todo.

1.1 Definição do problema

Até então, os algoritmos de EC apresentados têm como inspiração em algum ponto, conceitos da evolução Darwiniana. Entretanto, o processo evolutivo natural resulta da composição de diversos fatores. Os algoritmos evolutivos têm um grande foco nos aspectos genéticos da evolução, enquanto os algoritmos de inteligência coletiva e meméticos focam no aprendizado, tanto coletivo como individual.

Por outro lado, a teoria sintética da evolução [46] foi a conclusão de um longo trabalho que levou à conciliação das ideias evolutivas de Darwin por meio da seleção natural [18] e a herança biológica através da genética, proposta por Mendel [60]. Para isso, foi necessário o desenvolvimento da genética de populações, proposta principalmente por Fisher [84], Wright [98] e Haldane [38].

[1] Este conceito não deve ser confundido com *adaptativo* no sentido de refinamento de parâmetros.

[2] Tradicionalmente traduzido como inteligência de enxame. Aqui optamos por usar o termo inteligência coletiva devido ao fato de que nem todos os algoritmos apresentam um comportamento de enxame.

A genética de populações é o estudo da frequência e interação dos genes em uma população. A distribuição e mudança na frequência dos alelos ocorre em virtude dos processos evolutivos de seleção natural, deriva gênica, mutação e fluxo gênico. A composição de todos estes elementos na natureza é o que torna o mundo natural um dos mais complexos CAS, sendo a evolução o resultado final da interação de infinitos agentes em dinâmicas complexas, com auto-organização em múltiplos níveis. Como consequência, ocorre a variação fenotípica nas populações, levando aos fenômenos de especiação, adaptação, subdivisão e estruturação populacional [34, 75].

O funcionamento geral na EC é o resultado da inspiração biológica no processo evolutivo. Entretanto, tendo em vista que essa conexão é apenas uma inspiração, naturalmente existem diferenças em relação ao processo original, decorrentes de simplificações realizadas para que o processo computacional fosse possível de ser implementado e que a busca por soluções levasse a bons resultados.

Eiben & Smith [27] apresentaram uma extensa revisão bibliográfica com o foco na comparação entre os modelos de EC e o fenômeno natural da evolução. Algumas diferenças importantes são apresentadas na Tabela 1.1.

Os critérios comparados são o *fitness*, que é a medida utilizada para avaliar o quanto um indivíduo de uma população se encontra apto a diversas situações. A "seleção" é a ação que vai ocorrer sobre os indivíduos, fazendo com que continuem ou não vivos. O "mapeamento genótipo-fenótipo" define como os genes refletem no fenótipo dos indivíduos. A "variação" define como os genes são passados entre os indivíduos de uma população. O controle sobre a ação dos indivíduos é avaliado no critério "execução" e a "população" descreve como os indivíduos são estruturados coletivamente. De uma forma geral, vemos que as mudanças aplicadas em EC ocorrem no sentido de direcionar o caminho evolucionário para a resolução dos problemas para os quais se busca encontrar boas soluções. No entanto, uma maior aproximação do processo natural, e consequente aumento da plausibilidade biológica, é um um fator que merece atenção para estudos da computação natural.

Alguns estudos recentes apontam que as abordagens clássicas em computação evolutiva possuem uma forte inspiração no componente genético para a concepção dos modelos de computação evolutiva [59], ou até mesmo propõem o uso de conceitos de Ecologia e genética de populações para obter melhores resultados [71, 72]. Poucos trabalhos, no entanto, abordam a computação evolutiva realizando um estudo aprofundado da evolução, tratando a integração de algoritmos nos diferentes aspectos da teoria sintética da evolução. Ao ter o seu foco na seleção de genes baseada na avaliação direta da função de *fitness*, o processo resultante é o de direcionamento das características da população a um objetivo de interesse, similar à forma como ocorre a seleção de determinadas características em um acasalamento direcionado pela ação humana (por exemplo, em experimentos com ervilhas ou produção de animais na pecuária e animais domésticos).

Entretanto, uma característica da evolução na natureza é a ausência de um objetivo explícito. A evolução se dá como um processo de auto-organização emergente das relações complexas entre agentes do sistema natural. Por este motivo, pode ser encarado como sem um propósito específico (do inglês *Open-ended Evolution – OEE* [67]), produzindo continuamente elementos que podem eventualmente servir a determinados objetivos (as espécies). Invariavelmente, ao abarcar tais conceitos em um mesmo sistema, partimos em

Tabela 1.1: Principais diferenças entre evolução natural e algoritmos evolutivos (retirado e traduzido de [27]).

Evolução Natural	*Algoritmos Evolutivos*
Fitness	
Quantidade observada: efeito a posteriori da seleção e reprodução (no olho do observador)	Quantidade pré definida a priori que dirige a seleção e a reprodução
Seleção	
Complexa força de diversos fatores baseada em condições ambientais, outros indivíduos, da mesma espécie e de outras espécies (predadores). Viabilidade é testada continuamente; reprodutividade é testada em momentos discretos.	Operador aleatório com probabilidades de seleção baseada em um dado valor de *fitness*. Seleção de sobrevivência e de parentesco ocorrem em momentos discretos.
Mapeamento genótipo-fenótipo	
Processo bioquímico altamente complexo e desenvolvimentista, influenciado pelo ambiente.	Tipicamente uma simples transformação matemática ou procedimento parametrizado. Alguns sistemas utilizam mapas genótipo-fenótipo generativos e desenvolvimentistas.
Variação	
Prole criada a partir de um (reprodução assexuada), ou dois pais (reprodução sexuada). Transferência genética horizontal pode acumular genes de mais de um indivíduo.	Transferência genética vertical irrestrita. Prole pode ser gerada de qualquer número de pais: um, dois ou muitos.
Execução	
Execução paralela descentralizada.	Tipicamente centralizada, com mortes e nascimentos sincronizados.
População	
Enquadramento espacial implica populações estruturadas. Tamanho das populações variam de acordo com o número de nascimentos e mortes.	Tipicamente não estruturada e pan-mítica (todos os indivíduos são parceiros potenciais). Tamanho da população é usualmente mantido constante ao sincronizar o momento e o número de nascimentos e mortes.

direção aos requisitos para a construção de sistemas de vida artificial, e assim buscamos realizar a computação evolutiva ao mesmo tempo em que aumentamos sua plasibilidade biológica.

De forma a obter um sistema de vida artificial que alcance um nível mínimo de CAS,

caminhando em direção a OEE, e realizando computação evolutiva, o sistema deve possuir as seguintes características:

- O sistema deve comportar múltiplas populações para que diferentes espécies possam ser representadas. Como queremos também realizar computação evolutiva, as populações devem ser compostas por algoritmos dessa classe.
- As populações devem interagir por meio de relacionamentos ecológicos, proporcionando a complexidade necessária para um CAS.
- O número de indivíduos de uma população não deve ser estático. Para tanto, mortes e nascimentos devem ocorrer, fazendo com que o número de indivíduos flutue.
- Os agentes do sistema devem ser adaptativos, ou seja, capazes de responder a outros agentes e/ou ambiente em algum grau.
- O sistema deve proporcionar a transmissibilidade de características de um agente para um descendente direto, ou seja, reproduzir o fenômeno da hereditariedade quando ocorre uma reprodução e a produção de crias.

Assim, queremos explorar a realização de computação evolutiva, ao mesmo tempo que introduzindo os elementos de sistemas complexos da vida artificial, buscando mais uma direção de evolução sem um propósito ao invés de uma seleção direcionada. Pra tal fim, propomos que a seleção de indivíduos de EC tenha menos foco no valor da avaliação de *fitness*, em troca de interações que seriam consideradas relacionamentos ecológicos no mundo natural. Neste contexto, levantamos a hipótese de que essa abordagem pode realizar uma união de EC e AL passando pela SI.

A partir dos requisitos de ALife listados, apresentamos uma descrição mais detalhada dos objetivos pretendidos desta tese na próxima seção.

1.2 Objetivos

Em computação evolutiva existente uma ampla variedade de implementações, cada uma levando a um tipo diferente de algoritmo evolutivo. Apesar dos mais diversos modelos propostos e de combinações existentes [85], até onde pudemos averiguar, não existe um estudo que avalie o impacto evolutivo de uma etapa de seleção feita baseada em relacionamentos ecológicos. A partir dos modelos de vida artificial, é possível construir um sistema em que este estudo possa ser realizado.

Dessa forma, os objetivos deste trabalho consistem em especificar, implementar e validar um arcabouço de Vida Artificial para a realização de Computação inspirada pela natureza, mas que contenha as características que descrevemos para atingir um nível de sistema complexo adaptativo. Tal integração passa pela hipótese de que é necessária a retirada da etapa de seleção de indivíduos a partir de uma avaliação direta de uma função de *fitness*. Em seu lugar, a interação de indivíduos por meio de relacionamentos ecológico cumprirá esse papel.

Diante da construção de tal arcabouço, buscamos responder as seguintes questões:

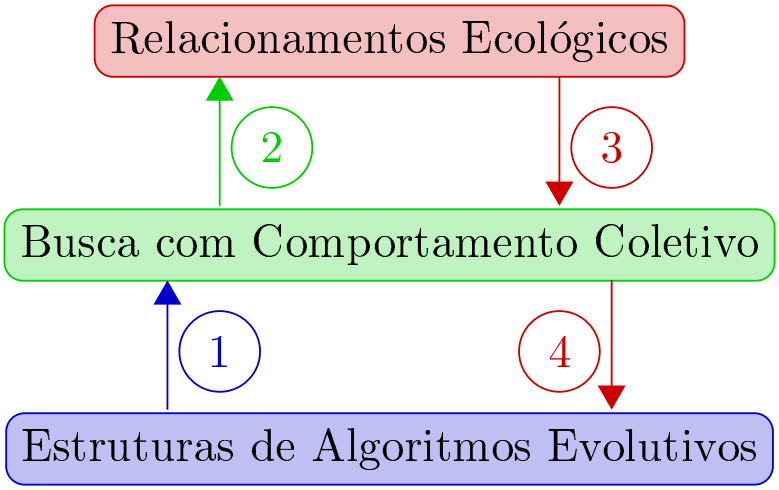


Figura 1.1: Visão conceitual do arcabouço EvoCycle.

Q-1 Quais trabalhos na literatura se relacionam com o tema proposto e servem como base para o a especificação do arcabouço?

Q-2 Ao substituir a seleção pela avaliação direta do *fitness* nos algoritmos de EC conseguimos gerar o fenômeno de adaptação?

Q-3 Ao realizar esse processo de computação evolutiva integrado ao de vida artificial por meio das relações ecológicas, o processo de busca continua válido?

Q-4 De que maneira podemos utilizar as informações produzidas nas simulações do modelo em benefício da AL e EC?

1.3 Proposta

Nesta tese, buscamos então responder tais questões propostas. Primeiramente foi realizado um extenso levantamento da bibliografia de Vida Artificial e Computação Evolutiva para responder a Q-1. De posse desse conhecimento, propomos o arcabouço de vida artificial EvoCycle. Com ele, abordamos as questões Q-2 e Q-3. No Evocycle, diferentes algoritmos de computação inspirada pela natureza representam espécies em um mesmo ambiente e o seu funcionamento é dividido conceitualmente em três camadas: genética, comportamento coletivo e relacionamentos ecológicos. Estas camadas são ilustradas na Figura 1.1.

Na camada genética estão algoritmos evolutivos que ditam o comportamento de indivíduos na segunda camada (seta 1). Na segunda camada temos populações compostas por algoritmos de inteligência coletiva. Uma vez que múltiplas populações passarão a ser empregadas em um mesmo processo de busca, surge a possibilidade da proposição de técnicas que utilizam a interação de populações em um nível superior ao de comportamento coletivo, neste caso, as relações ecológicas (seta 2). As interações nos níveis ecológicos são responsáveis pela seleção de indivíduos no nível coletivo (seta 3), que por sua vez resulta na criação de novos indivíduos com material genético apenas de indivíduos sobreviventes das interações (seta 4). Fecha-se assim um ciclo para o processo evolutivo.

Somado a isso, o arcabouço incorpora elementos de vida artificial, nos quais os agentes se reproduzem ou morrem baseados no resultado da busca por recursos, de forma a proporcionar o surgimento de dinâmicas populacionais. Como propomos que no arcabouço exista uma integração entre a codificação genética e o comportamentos de busca dos algoritmos de EC, o comportamento de busca pode evoluir de diferentes formas ao longo do tempo, possibilitando que a seleção genética não seja feita de forma direta.

Com isso, foi possível realizar um estudo da integração de modelos de algoritmos evolutivos, inteligência coletiva e sistemas ecológicos a partir dos conceitos de vida artificial e avaliar a adaptação dos indivíduos com a devida instanciação do arcabouço, aplicados no cenário de otimização contínua, com o uso de funções de *benchmark* tipicamente utilizadas para avaliar algoritmos bio-inspirados.

Para responder a segunda questão então, avaliamos a variação das frequências genéticas em populações presentes no EvoCycle com o intuito de certificar o arcabouço como um sistema válido de vida artificial que leva a um comportamento de busca como resultado do processo de evolução sem um propósito (OEE). Ao mesmo tempo, com esse processo de busca, respondemos a terceira questão, verificando a ocorrência da capacidade de otimização.

Finalmente, para responder a questão Q-4, propomos um estudo da resiliência em populações do EvoCycle. O objetivo é investigar novas métricas e informações que possam ser empregadas em cenários de simulação de vida artificial e em cenários de computação evolutiva.

1.4 Organização da tese

No restante desta tese, apresentamos o arcabouço EvoCycle e a partir de sua implementação buscamos responder as questões de pesquisa apresentadas através da realização de simulações de vida artificial empregadas em problemas de otimização. Para isso, apresentamos uma revisão da literatura no Capítulo 2, no qual apresentamos os conceitos de vida artificial, os algoritmos de computação evolutiva, bem como os pontos que enxergamos de conexão entre ambos. No Capítulo 3, fazemos a introdução do nosso arcabouço, apresentando seu algoritmo e uma implementação preliminar para validação do mesmo. No Capítulo 4, expandimos a implementação com a inclusão de mais de uma espécie para a formação de um ecossistema e a realização de predação entre as duas. A partir dos resultados das dinâmicas populacionais obtidas nas simulações, propomos um estudo de resiliência das populações de predadores no Capítulo 5. Por fim, apresentamos as principais contribuições do trabalho e possíveis trabalhos futuros no Capítulo 6.

Capítulo 2

Trabalhos Relacionados

Estudamos a literatura de computação bio-inspirada para compreender como os aspectos biológicos foram representados em diferentes técnicas e como evoluíram ao passo que avanços computacionais foram sendo alcançados. Da mesma forma, nos aprofundamos na literatura de vida artificial para identificar elementos que poderiam ser incorporados ao sistemas de computação evolutiva, de forma a aproximar as representações computacionais dos fenômenos biológicos de suas inspirações naturais.

A partir do conhecimento das técnicas que podem ser empregadas em cada uma das camadas do EvoCycle, o próximo passo consiste em construir meta-heurísticas que apresentem tanto comportamento coletivo quanto de hereditariedade, para que com estes dois elementos, possam evoluir com o tempo. Então verificamos as maneiras com que a integração entre os aspectos genéticos e de comportamento coletivos poderia ser realizada.

Em seguida é necessário estabelecer as formas que diferentes populações de meta-heurísticas podem se relacionar. Tendo o objetivo de nos aproximarmos do sistema natural com o uso de populações de EC, priorizamos abordagens que buscam construir ecossistemas.

Sendo assim, apresentamos neste capítulo um panorama dos levantamentos teóricos na Seção 2.1. Muitos trabalhos se beneficiaram da combinação entre essas as técnicas de algoritmos evolutivos e de inteligência coletiva. Estes trabalhos são apresentados na Seção 2.2. Por fim, a Seção 2.3 apresenta trabalhos que incorporam estratégias do campo da Ecologia.

2.1 Fundamentos Teóricos

Começamos com o levantamento da computação evolutiva, agrupando trabalhos de acordo com o as camadas propostas no EvoCycle. Na Seção 2.1.1, os principais trabalhos dos primórdios da computação evolutiva são apresentados, sendo esta seção responsável por categorizar algoritmos que tomaram como inspiração aspectos relacionados à genética. A Seção 2.1.2 apresenta algoritmos que usam como inspiração o comportamento social populacional e apresenta trabalhos realizados com o comportamento coletivo. Depois, apresentamos um levantamento do campo da vida artificial na Seção 2.1.3.

2.1.1 Algoritmos Evolutivos

Os algoritmos evolutivos (do inglês *Evolutionary Algorithms* – EA) surgiram no contexto do uso da programação de computadores para a obtenção de soluções numéricas para problemas difíceis, que ao serem tratados por abordagens precisas, como as analíticas, a busca por soluções se torna inviável.

Sendo uma heurística uma forma de encontrar uma solução boa o suficiente para um problema, mas sem a garantia de produzir uma solução ótima (a melhor possível), EA são considerados meta-heurísticas (MH) por produzirem elementos que guiam o processo de busca por soluções [11]. No caso específico dos EA, os elementos que guiam o processo de busca se inspiram no processo evolutivo do mundo natural.

A primeira das técnicas desenvolvidas foi a Programação Evolutiva (do inglês *Evolutionary Programming – EP*) [31], por Fogel em 1966. A EP utiliza o conceito de máquinas de estado finito (do inglês *Finite State Machines – FSM*) para representar uma solução. Uma vez exposta a um conjunto de símbolos, uma FSM é avaliada a partir da tentativa de predição do próximo símbolo. Para tal avaliação, pode-se utilizar o erro da predição. Para a geração de novas FSMs na população, alterações de estado são obtidas por meio de mutações nas FSMs, de acordo com uma distribuição de probabilidade pré-estabelecida.

Em 1973, foi proposto o trabalho de Rechenberg denominado estratégias evolutivas (do inglês *Evolutionary Strategies – ES*) [10], que seria complementado por Schwefel em 1977. As soluções nas ESs são representadas por vetores de pontos em um espaço de busca, mas o que caracteriza as ESs é o fato de cada indivíduo ser associado a parâmetros de autoadaptação, que posteriormente são utilizados para modificar o vetor de pontos. Inicialmente, as ESs apresentavam apenas um indivíduo por geração, e apenas um indivíduo era obtido a partir de mutação. A geração seguinte seria a escolha do que possui maior *fitness* entre os dois. Posteriormente, diferentes ESs foram definidas em que o conceito de população é utilizado, podendo variar quantos indivíduos são utilizados para gerar filhos, inclusive utilizando recombinação; e podendo variar quantos indivíduos são selecionados para a próxima geração.

Desenvolvidos por Holland em 1975, nos Algoritmos Genéticos (do inglês *Genetic Algorithms – GA*) [43], as soluções são representadas por uma cadeia de bits. A seleção de indivíduos ocorre probabilisticamente com uma combinação entre a proporção do *fitness* e uma função de escolha baseada em roleta. Os indivíduos selecionados sofrem mutações aleatórias, sendo que cada posição da cadeia tem uma probabilidade de mutação independente das demais. Além disso, é utilizado o operador de recombinação, no qual sub-cadeias de dois indivíduos são trocadas.

Finalmente, a programação genética (do inglês *Genetic Programming – GP*) [51] foi popularizada por Koza em 1992. A GP surge com a ideia de evoluir códigos de programação. Como um programa pode ser organizado sintaticamente por uma árvore, esta é a representação típica de uma solução em GP. Os indivíduos da população são selecionados proporcionalmente ao *fitness* e a variação ocorre a partir dos operadores genéticos.

Estes quatro algoritmos são considerados os algoritmos clássicos de computação evolutiva e compreendem uma grande área em que diversas aplicações e variações foram realizadas. Um outro importante algoritmo que vale a pena ser mencionado é referente

a *Differential Evolution - DE* [86]. Proposto em 1996 por Storn e Price, a ideia geral da DE é que a partir de um conjunto de agentes novos agentes são formados por meio do uso de uma equação fixa. Caso possua melhor fitness o agente é mantido na população, do contrário é descartado. Uma revisão sobre o tema pode ser encontrada em [20].

Programação Genética

Devido ao posterior uso da GP dentro desta tese, faremos uma descrição mais detalhada deste método. Como apresentado previamente, a GP foi idealizada com o intuito de evoluir programas de computadores. Inspirada pelo processo natural da evolução da Biologia, cada programa é estruturado como uma árvore, que funciona como o seu código genético.

Cada programa (indivíduo) é avaliado por uma função de *fitness*. Os melhores indivíduos são escolhidos para se reproduzirem e novos indivíduos são produzidos por este processo. A variação é introduzida pela aplicação de operadores genéticos como mutação e recombinação. Este processo é repetido diversas vezes, de forma a criar uma nova geração de indivíduos a cada vez que indivíduos são selecionados e as variações são aplicadas. Ao final do processo é esperado que indivíduos com bom *fitness* sejam encontrados. Uma revisão dos usos da GP foi realizada por De Lorenzo et al. e pode ser encontrada em [23]. Um pseudo código para a realização de computação evolutiva pode ser encontrado no Algoritmo 1.

Algoritmo 1 Programação Genética

```
1:  função GP(Internos, Folhas)
2:      POP ⟵ ÁrvoreAleatória(Internos, Folhas)
3:      enquanto Não satisfeito faça
4:          para todo  i ∈ POP faça i.fitness ⟵ Avaliar(i)
5:          fim para
6:          melhores ⟵ Selecionar(POP)
7:          filhos ⟵ Recombinar(melhores)
8:          POP ⟵ melhores + filhos
9:      fim enquanto
         devolve Melhor indivíduo i
10: fim função
```

A execução do Algoritmo 1 se dá então com a inicialização da população de programas (POP) gerada aleatoriamente a partir dos operadores internos (Internos) e terminais (Folhas) que configuram um determinado problema. Em seguida, todos os indivíduos (i) são avaliados, obtendo-se um valor de *fitness*. Os melhores são selecionados e novos programas são gerados para compor a nova população. Os novos indivíduos são gerados a partir da recombinação de dois indivíduos selecionados. Este processo é repetido até um determinado critério de parada. Por fim, o melhor indivíduo encontrado em todo o

processo é retornado. Para que essa execução ocorra, algumas outras caracterizações dos elementos que compõem a GP são necessárias. Os principais elementos são:

- **Representação**: usualmente é utilizada a estrutura de árvore, que representa os programas hierarquicamente estruturados. Neste sentido, as representações variam de acordo com o número de filhos (representações que podem ser, por exemplo, binária, ternária, N-ária) e levando-se em conta um limiar de profundidade máxima. Os operadores internos podem ser compostos por operadores matemáticos ($+, -, \times, \backslash$), funções matemáticas (seno, cosseno, exponencial, raiz), operadores booleanos (E, OU, NÃO), condicionais (SE), repetidores (ENQUANTO) e funções específicas de cada domínio. Os nós terminais são compostos por valores referentes ao problema tratado.

- **Geradores:** As árvores da GP são geradas aleatoriamente. Inicialmente é escolhido um nó interno do conjunto de operadores como raiz e para cada um de suas ramificações, são escolhidos ou uma nova função ou um nó terminal. Caso um nó terminal seja escolhido o crescimento da árvore é interrompido. Caso uma nova função seja escolhida, o processo continua recursivamente. Essa maneira de criação da árvore ainda é dividida em três formas de acordo com a implementação: **Full**, na qual todas as folhas têm distâncias igual ao máximo especificado; **Grow**, na qual ao menos uma folha tem distância igual ao máximo especificado, porém com formatos variados; e finalmente a **ramped-half-and-half**, na qual é feita uma combinação dos dois últimos, e a população é criada com alturas variando entre 2 e o máximo especificado, sendo para cada valor metade gerada pelo método *Full* e metade pelo *Grow*.

- **Seleção**: os indivíduos são usualmente avaliados por funções de *fitness* e a seleção pode ser realizada de diferentes formas: elitismo, selecionando indivíduos pelo valor direto do fitness, no qual por meio de um ranqueamento, indivíduos são selecionados após uma ordenação de toda a população em relação ao *fitness*; Torneio de seleção, no qual um grupo de indivíduos é escolhido da população a partir de um pareamento dois a dois, sempre sendo escolhido o de maior *fitness*, como chaves em um torneio; roleta, no qual os indivíduos com maior *fitness* tem maiores chances de serem escolhidos em um sorteio.

- **Recombinação**: sub-árvores aleatórias de dois indivíduos são selecionadas e trocadas. A escolha dos pais é feita tipicamente usando os mesmos métodos de seleção para a próxima rodada. Como os indivíduos possuem tamanhos diferentes, um ponto em cada árvore é escolhido aleatoriamente com uma distribuição uniforme. A recombinação é feita utilizando este ponto como a raiz de uma sub-árvore para cada pai, e por fim estas subárvores são trocadas para a geração de dois novos filhos.

- **Mutação**: usualmente implementada a partir da poda de uma sub-árvore de um indivíduo, nesta operação, gera-se uma sub-árvore aleatória para substituir a parte podada.

2.1.2 Inteligência Coletiva

A computação natural não se restringe aos algoritmos evolutivos. Dentro da categoria de meta-heurísticas, também podem ser especificadas aquelas que tomam como inspiração processos que ocorrem na natureza para a construção de algoritmos, os chamados algoritmos bio-inspirados [24].

Surge na década de 1990 a inteligência a partir de bandos ou enxames (do inglês *Swarm Intelligence – SI*). Nestes sistemas, ao invés de informação genética, em que a melhora das soluções ocorre por meio de operadores genéticos, soluções melhores são encontradas a partir da interação entre os elementos de um bando (*Swarm*). O comportamento individual de cada elemento não necessariamente busca otimização, e as interações fazem com que ocorra um comportamento emergente coletivo, que leva a melhores soluções.

Os dois principais algoritmos de SI são a otimização colônia de formigas (ACO) [25] e e otimização bando de partículas (PSO) [26]. Proposto em 1995 por Kennedy e Eberhart, o PSO tem sua inspiração na movimentação de bandos de pássaros, cardumes de peixes e teoria de enxames em geral. Nele, partículas se movimentam de forma coordenada pelo hiperespaço de soluções tentando encontrar um valor ótimo.

O ACO, proposto em 1996 por Dorigo, se baseia nos caminhos criados por formigas para encontrar comida. Cada formiga então é considerada um elemento de busca. Inicialmente, cada formiga da população tem um movimento aleatório no espaço de soluções e cada uma delas deixa em seu caminho um feromônio. Quanto maior a concentração de feromônio em um caminho, mais formigas serão atraídas para este caminho.

Baseando-se na inspiração do comportamento coletivo, diversos outros algoritmos foram propostos. Citamos algumas técnicas mais recentes para ilustrar que as mais diversas fontes de inspiração biológica foram utilizadas nessa classe de algoritmos e a emergência proveniente da interação entre agentes surge de diferentes formas, como por exemplo o otimizador baseado em biogeografia [83], otimizador lobo cinza [62] e o otimizador libélula (Dragonfly) [61].

O primeiro se baseia no conceito de que populações aumentam e diminuem de acordo com as condições que um habitat proporciona aos seus residentes. Partindo deste conceito, migrações podem ocorrer, fazendo com que o habitat possa ou não suportar a quantidade de indivíduos presentes. Assim, a busca ocorre por um habitat com alto nível de capacidade (melhores soluções). No segundo, o comportamento de caça dos lobos cinza é simulado em partículas, sendo que as três melhores posições encontradas definem o status de lobos que irão liderar o bando, os lobos alfa, beta e gama, respectivamente. Os piores lobos têm a função de realizar buscas aleatórias enquanto que os intermediários seguem o comportamento dos três melhores Finalmente, no terceiro, é considerado como libélulas procuram por alimento. Neste modelo, também é incorporada a estratégia que as mesmas utilizam para escapar de predadores.

Para mais informações sobre os algoritmos de inteligência coletiva, um *survey* destas técnicas pode ser consultado em [69]. Assim como fizemos na seção anterior, apresentamos a seguir mais detalhes acerca das meta-heurísticas que foram empregadas nesta tese: PSO e o ABC.

Particle Swarm Optimization – PSO

O PSO [26] é uma meta heurística que toma como inspiração o padrão de movimento coletivo de grandes grupos de animais, como bando de pássaros e cardume de peixes. O comportamento no qual todos se movem em conjunto, de forma coordenada, é utilizado para a busca de soluções ótimas.

Em sua especificação, múltiplas soluções são consideradas partículas em movimento, formando uma população. A cada partícula, é atribuída uma certa velocidade aleatória. O processo de movimentação é dividido em iterações no qual, a cada iteração as velocidades das partículas são modificadas levando em consideração a melhor posição entre todas as partículas e a melhor posição encontrada por cada uma delas, segundo um critério de *fitness*.

Uma distinção dos algoritmos evolutivos é que o PSO tem a característica de preservar o conhecimento adquirido ao longo das iterações, propriedade que não é necessariamente respeitada pelos algoritmos evolutivos clássicos, uma vez que a melhor solução pode não ser propagada para gerações seguintes.

Para simular esse movimento coordenado e baseado nas melhores posições globais e locais, as partículas se movimentam pelo espaço de acordo com as seguintes equações:

$$\vec{V}_{i+1} = \vec{V}_i + R_1 \times \phi_1 \times (\vec{P}_{best_i} - \vec{X}_i) + R_2 \times \phi_2 \times (\vec{G}_{best} - \vec{X}_i) \tag{2.1}$$

$$\vec{X}_{i+1} = \vec{X}_i + V_i \vec{+} 1 \tag{2.2}$$

em que X representa a posição de cada partícula no espaço em um dado momento no tempo i, V representa uma função de velocidade que varia de acordo com o valor prévio da velocidade da partícula, a melhor posição que a partícula visitou (P_{best}) e a melhor posição encontrada pelo bando (G_{best}). ϕ_1 e ϕ_2 são os coeficientes de aceleração, que aliados aos fatores aleatórios, R_1 e R_2, representam o quanto o conhecimento social e o conhecimento individual tem influência no movimento. De forma geral, os passos do PSO em sua versão original podem ser resumidos seguindo o Algoritmo 2.

No Algoritmo 2, uma população POP é composta por N partículas $\vec{X}_i$, inicialmente posicionadas aleatoriamente em uma posição do $\Re^{Dim}$, sendo Dim o número de dimensões tratado, com uma velocidade $\vec{V}_i$. As partículas são avaliadas na função objetivo F e se deslocam pelo espaço seguindo as Equações 2.1 e 2.2. Ao final o melhor posicionamento é retornado em $\vec{G}_{best}$.

Esta forma clássica do PSO foi extensivamente estudada e diversas modificações foram propostas. Para uma revisão de propostas existentes, o leitor pode se referir aos trabalho de Zhang et al. [104] e Houssein et al. [45].

Artificial Bee Colony – ABC

Outro importante algoritmo de inteligência coletiva é o *Artificial Bee Colony* (ABC) [49]. ABC é inspirado na estratégia empregada por abelhas ao buscar fontes de alimento e ao construir colmeias em "boas" posições. O algoritmo pode ser dividido em três etapas: a etapa das abelhas trabalhadoras, a etapa das abelhas espectadoras, e a etapa das abelhas

Algoritmo 2 *Particle Swarm Optimization*

função PSO($\phi_1, \phi_2, N, Dim, F$)
  POP $\longleftarrow$ [(X_1, V_1), ..., (X_N, V_N)]
  enquanto Não satisfeito **faça**
    para todo $\vec{X}_i \in POP \mid \vec{X}_i \in \Re^{Dim}$ **faça** X_i.fitness $\longleftarrow$ $F(X_i)$
      se X_i.fitness $<$ P_{best_i}.fitness **então**P_{best_i}.fitness $\longleftarrow$ X_i.fitness $P_{best_i} \longleftarrow X_i$
      fim se
      se X_i.fitness $<$ G_{best}.fitness **então** G_{best}.fitness $\longleftarrow$ X_i.fitness $G_{best} \longleftarrow X_i$
      fim se
      $\vec{V}_{i+1} \longleftarrow \vec{V}_i + R_1 \times \phi_1 \times (\vec{P}_{best_i} - \vec{X}_i) + R_2 \times \phi_2 \times (\vec{G}_{best} - \vec{X}_i)$
      $\vec{X}_{i+1} \longleftarrow \vec{X}_i + \vec{V}_{i+1}$
    fim para
  fim enquanto
  devolve $\vec{G}_{best}$
fim função

batedoras.

Inicialmente, um certo número de colmeias é posicionada aleatoriamente no espaço de busca. Então, na etapa das abelhas trabalhadoras, uma abelha visita uma das colmeias. Assim, para cada colmeia na população, uma nova posição de colmeia é criada pela modificação de uma das coordenadas da colmeia melhor posicionada. Essa modificação vem da soma ponderada da melhor posição e da diferença entre a colmeia sendo visitada atualmente e uma selecionada aleatoriamente, de acordo com a seguinte equação:

$$v_{ij} = x_{ij} + \phi_{ij} \times (x_{ij} - x_{kj}) \tag{2.3}$$

em que v_i é a nova colmeia i gerada e j é uma dimensão dessa colmeia escolhida aleatoriamente, x_i é a colmeia sendo visitada atualmente por i, e x_k é uma colmeia selecionada aleatoriamente, diferente da visitada por i. ϕ é um número aleatório entre -1 e 1. Após a modificação, se a colmeia criada tiver um *fitness* melhor que o da colmeia sendo visitada atualmente, esta nova colmeia substitui a antiga na população.

Em seguida, acontece a etapa das abelhas expectadoras. Esta etapa segue um processo similar, com a diferença que ao invés de modificações serem realizadas em uma colmeia por vez, a mudança ocorre em uma colmeia selecionada com proporção ao *fitness* que possui, de acordo com:

$$p_i = \frac{fit_i}{\sum_{N=1}^{SN} fit_n} \tag{2.4}$$

em que p_i é a probabilidade de uma colmeia ser escolhida para ser visitada, dado o seu *fitness* fit_i em relação a soma do *fitness* de todas as colmeias. A ideia desta etapa é que após a etapa anterior, as expectadoras observaram o resultado das abelhas trabalhadoras

e visitam as colmeias com maior potencial[1].

O último passo é o das abelhas batedoras, que são responsáveis por verificar se as colmeias foram visitadas um determinado número de vezes e não foi movida para uma região melhor. Se isso é verdade para uma colmeia em particular, essa colmeia é movida para uma região aleatória.

A cada iteração é atribuída um nível de energia para cada colmeia, correspondente à posição que cada uma delas se encontra, seguindo a equação:

$$E_i = \frac{1}{|fit + 1|} \tag{2.5}$$

em que fit é o *fitness* correspondente da posição que a colmeia i ocupa em uma dada iteração. O Algoritmo 3 sumariza os principais passos do ABC.

[1] A inspiração biológica é que as abelhas comunicam essa informação por meio de movimentações que o autor denomina de dança das abelhas.

Algoritmo 3 *Artificial Bee Colony*

procedimento ABC(ϕ_1, l, N, Dim, F)
  SN $\longleftarrow \frac{N}{2}$
  POP $\longleftarrow [\vec{X_1}, ..., \vec{X_{SN}} \mid \vec{X_i} \in \Re^{Dim}]$
  enquanto Não satisfeito **faça**
    para todo $\vec{X_i} \in POP$ **faça**
      $V_{ij} \longleftarrow X_{ij} + \phi_{ij} \times (X_{ij} - X_{kj})$
      V_i.fitness $\longleftarrow F(\vec{X_i})$
      se V_i.fitness $<$ X_i.fitness **então**
        X_i.fitness $\longleftarrow$ V_i.fitness
        $X_{ij} \longleftarrow V_{ij}$
        X_i.tentativas $\longleftarrow 0$
      senão
        X_i.tentativas $\longleftarrow$ X_i.tentativas + 1
      fim se
    fim para
    para 1 até SN **faça**
      $P(i) \longleftarrow \frac{X_i.fitness}{\sum_{n=1}^{SN} X_i.fitness}$
      $V_{ij} \longleftarrow X_{ij} + \phi_{ij} \times (X_{ij} - X_{kj})$
      V_i.fitness $\longleftarrow F(X_i)$
      se V_i.fitness $<$ X_i.fitness **então**
        X_i.fitness $\longleftarrow$ V_i.fitness
        $X_{ij} \longleftarrow V_{ij}$
        X_i.tentativas $\longleftarrow 0$
      senão
        X_i.tentativas $\longleftarrow$ X_i.tentativas + 1
      fim se
    fim para
    para todo $X_i \in POP$ **faça**
      se X_i.tentativas $>$ l **então**
        $\vec{X_i} \longleftarrow [\vec{X} \in \Re^{Dim}]$
        sair para
      fim se
    fim para
  fim enquanto
fim procedimento

No Algoritmo 3, SN representa o número total de colmeias, que serão posicionadas aleatoriamente nas posições $\vec{X_i}$ com coordenadas pertencentes ao $\Re^{Dim}$, compondo o conjunto POP. Para cada colmeia em POP, a fase das abelhas trabalhadoras ocorre entre

as linhas 5 e 15, a das abelhas espectadoras entre as linhas 16 e 27, e a etapa das abelhas batedoras entre as linhas 28 e 33.

2.1.3 Vida Artificial

O termo Vida Artificial refere-se à possibilidade da criação de fenômenos biológicos pelo ser humano ao invés da Natureza, como apontado por Christopher G. Langton em seu resumo sobre Vida Artificial [53]. Vida Artificial é o campo responsável pelo estudo científico das implicações dessa criação em diversos segmentos, como tecnologia, filosofia e artes.

A importância do estudo de sistemas de vida artificial reside no apoio que pode proporcionar ao campo da Biologia, campo científico do estudo da vida. Para realizar o estudo da vida, são necessários dados provenientes da observação do mundo natural. Muitas vezes existem grandes dificuldades na aquisição destes dados, sendo por limitações financeiras, éticas ou até mesmo de tempo. Por este motivo, a reprodução dos elementos naturais de maneira artificial pode auxiliar na realização do estudo de fenômenos que os envolvem.

Por exemplo, o estudo do funcionamento de um determinado órgão pode envolver alto custo para aquisição ou um compromisso ético pelo sacrifício de seres vivos. Outro exemplo seria o estudo de um fenômeno biológico que necessitaria de uma longa escala temporal para ser visualizado. Assim, a vida artificial poderia ser empregada para validar conceitos ou testar hipóteses a partir de simulações de fenômenos naturais.

Além disso, a vida artificial pode proporcionar elementos para expandir o estudo da Biologia às formas de vida diferentes de como a conhecemos. Uma vez que as formas de vida do mundo natural compartilham as mesmas condições de existência e surgimento, teorias para o funcionamento ou surgimento de vida em diferentes condições das encontradas aqui podem ser realizadas com soluções baseadas em sistemas de vida artificial.

Assim, este tipo de sistema pode auxiliar na dedução de regras gerais ou propriedades que são compartilhadas por qualquer sistema de vida e diferenciar princípios ou propriedades que são específicos de uma forma de vida, devido às particularidades do sistema local. Na falta de outros exemplos de vida, uma opção seria criar formas de vida alternativas.

A reprodução dos fenômenos biológicos pode então ser produzida de diferentes formas, como visto em [53]. Quando simulações computacionais são utilizadas, a disciplina é vista como *soft*, quando a construção de maquinários é utilizada chamamos de *hard*, e quando reações químicas ou material biológico são empregados, então denominamos *wet*. A vida artificial *hard* e *wet* são objetos de estudo da bioquímica e robótica respectivamente, e uma desvantagem em relação à versão *soft* é que demandam uma quantidade maior de recursos e conhecimento.

Neste trabalho, temos o nosso foco em sistemas computacionais que buscam reproduzir ou criar fenômenos biológicos por meio de softwares.

Características de Interesse

Nesta tese, estamos interessados em uma simulação de Vida Artificial com o uso de ecossistemas. Muitos trabalhos seguiram essa mesma linha para estudar diferentes aspectos de ecossistemas. Uma inspiração inicial é o sistema Echo [32], que foi implementado com o

intuito de validar a teoria de sistemas complexos adaptativos (CAS) [44]. O sistema Echo é definido como complexo uma vez que é composto por agentes que interagem entre si e produzem padrões emergentes, com os agentes e interações evoluindo ao longo do tempo. O sistema é composto por regiões onde os agentes são alocados, produzindo recursos de acordo com parâmetros e se relacionando com outros agentes. Em cada região, os agentes podem trocar recursos, lutar uns com os outros e/ou se reproduzirem como definidos por seus códigos genéticos. Outras simulações de ecossistemas de Vida Artificial incluem o sistema Tierra [90] e o sistema Avida [2], nos quais agentes são autômatos que basicamente trabalham como programas de computadores disputando recursos de um computador.

Relações ecológicas específicas também foram abordadas em diferentes sistemas. O relacionamento presa-predador em específico, um de nossos interesses nesse trabalho, é um que pode ser citado. Gras et al. [35] propuseram um sistema presa-predador baseado em agentes chamado Ecosim. Nele, são usados *Fuzzy Cognitive Maps* (FCM) para simular o comportamento de agentes. Com um mecanismo de percepção de distância, os agentes apresentam o comportamento de evasão e busca por alimento. Na etapa de reprodução, valores das respectivas FCM são trocados. Dentre os principais resultados do estudo, destaca-se a diferenciação de espécies ao longo das simulações.

Alfonseca e Gil [4] evoluíram um ecossistema de expressões matemáticas utilizando GE. Ao realizar a evolução de novas expressões, este sistema não leva em conta características espaciais associadas aos indivíduos do sistema. Estes indivíduos são avaliados seguindo uma função de *fitness* e dinâmicas de presa-predador são produzidas na população.

Um ecossistema de modelos 3D foi proposto por Ito et al. [48]. Nesse sistema, os agentes possuem atributos físicos que desempenham um papel importante na movimentação e refletem no *fitness* de cada um. Algoritmos Genéticos são usados para evoluir as características de cada população, separadamente da interação entre os agentes. Padrões clássicos de predação foram observados e os autores notaram uma correlação entre o volume dos corpos na ausência ou presença de predadores.

Adami et al. [3] também apresentaram um ecossistema de presa-predadores em um *grid* bi-dimensional. Cada agente tem o seu campo de visão modelado por uma cadeia de Markov, correspondendo ao seu código genético. A cadeia é evoluída por uma GA em etapas, separadas da interação das populações. Os resultados obtidos demostraram que, uma vez que as presas aprendem um comportamento de agrupamento, predadores se beneficiam de um campo de visão mais focado. Mas quando isso acontece, as presas começam a apresentar um comportamento de dispersão, o que por sua vez resulta em predadores com campo de visão mais amplo.

Com o objetivo de construir um sistema nos quais agentes de um bando pudessem atingir *open ended evolution* (OEE), Witkowski e Ikegami [97] propuseram um sistema utilizando *boids* que têm seu comportamento estabelecidos por redes neurais artificiais (ANN). As configurações das NN evoluem com o tempo utilizando GA. Suas principais descobertas foram que para se obter OEE o aspecto da descentralização é muito importante, assim como a comunicação entre os agentes. Três aplicações foram apresentadas, uma no cenário de busca, outra no dilema do prisioneiro e na teoria dos jogos e uma última relativa à escalabilidade.

Sunehag et al. [87] desenvolveram um ecossistema presa-predador usando agentes com

modelos de aprendizado de reforço para se mover em espaços 2D e 3D. Uma de suas maiores contribuições foi a apresentação de um ambiente de simulação genérico com 3 níveis tróficos. Nos seus achados, foi possível observar o surgimento de dinâmicas populacionais e comportamento de enxame, assim como diferentes padrões de interações entre as espécies.

Dos sistemas listados elegemos então as seguintes propriedades como características de interesse para serem incorporadas ao EvoCyle:

- **Recursos:** propriedade do sistema que especifica um elemento necessário aos agentes para a realização de alguma atividade.
- **Tempo:** propriedade do sistema que determina a duração de uma simulação em unidades.
- **Energia:** propriedade que quantifica os recursos que foram adquiridos por cada agente no sistema.
- **Idade:** propriedade que quantifica as unidades de tempo que um determinado agente viveu no sistema artificial.
- **Tamanho da população:** medida do número de agentes que varia de acordo com mortes e nascimentos.

2.2 Hibridizações de Meta-heurísticas

Apesar de seguir uma mesma linha geral, as áreas de Algoritmos Evolutivos e Inteligência Coletiva abordam os conceitos de otimização e aprendizado sob diferentes perspectivas. Nos algoritmos evolutivos, a evolução por meio da seleção leva a uma adaptação que produz melhores soluções. Na inteligência coletiva, a interação entre todos os elementos do grupo faz emergir um comportamento que leva às melhores soluções coletivamente.

Necessitamos agora de uma maneira de combinar estas duas características, de forma que em uma população seja possível identificar um tipo de comportamento, como o comportamento coletivo, e que este comportamento seja o resultado da expressão de subestruturas, como os genes. Antes de tratarmos as combinações diretamente, damos atenção à questão do comportamento individual na forma de aprendizado.

A combinação de MHs com outras técnicas de otimização (inclusive outra MH) é o que configura uma Meta-heurística Híbrida. Diferentes formas de realizar a combinação destas técnicas podem ser encontradas na literatura. Embora exista um grande número de MHs bio-inspiradas, apresentamos um levantamento baseado principalmente no PSO, devido ao seu uso nesta tese. No entanto, também podemos encarar as formas como MHs populacionais podem ser combinadas de uma forma genérica a partir disso.

Este levantamento de hibridizações é apresentado a seguir. Na Seção 2.2.1, é abordada a relação do comportamento e aprendizado, bem como representações destes conceitos em MHs; na Seção 2.2.2, são apresentadas as técnicas que fazem uso do aprendizado para o aprimoramento das MHs. Exemplos de hibridizações envolvendo uma MH do tipo

de SI e outra do tipo EA são apresentados na Seção 2.2.3. Por fim, tendo em vista o grande número de hibridizações presentes na literatura, categorizações das combinações são apresentadas na Seção 2.2.4.

2.2.1 Comportamento: social × individual

Os mecanismos pelos quais o processo de busca das MHs de SI ocorre podem ser diversos. Um fator importante para o nosso trabalho é a relação entre o comportamento desenvolvido por um indivíduo, configurado como fenótipo e o comportamento derivado da população como um todo, comportamento coletivo emergente. Somado a isso, existe o papel que a informação adquirida por um indivíduo, na forma de aprendizado, desempenha no comportamento geral. Abordamos então a questão do aprendizado no contexto das MHs de EC.

Em seu trabalho de 1989, Hinton et al. [41] discutiram a hipótese de que um indivíduo realiza aprendizado ao longo da sua vida, e tal aprendizado é útil para o processo de evolução, mesmo que tal conhecimento não seja propagado para a configuração genética. Para tanto, é realizada a avaliação da frequência de genes alterados em boas soluções de um algoritmo genético com e sem o aprendizado de uma rede neural artificial.

Em trabalhos mais recentes, é possível ver que esta ainda é uma questão que necessita de maior exploração. Enquanto em [68] e em [12], os autores afirmam que altos valores de adaptação individual são atingidos com baixo nível de adaptação populacional e vice-versa, em [17], os autores evidenciam a capacidade de melhora. Ainda assim, em todos estes estudos, são apresentados cenários em que relacionar o fenótipo e o genótipo levam a melhores resultados. De forma geral, isso ocorre em cenários multi-objetivo.

Muitas são as maneiras em que o conceito de aprendizado pode ser usado para ampliar a eficácia de MHs otimizadoras. Além disso, os algoritmos apresentados até o momento não necessitam de informações específicas sobre o problema que desejam encontrar soluções. Esta é uma grande vantagem destes algoritmos. No entanto, considerando a possibilidade de usar conhecimento específico das aplicações, não poderíamos deixar de mencionar toda uma categoria de MHs que trabalha com aprendizado. Essa categoria é a dos algoritmos meméticos (MA) [64], que se inspiram no aprendizado cultural.

O conceito de Meme, definido pelo filósofo teórico Richard Dawkings, é de uma subunidade do conhecimento humano [21]. Dentro de uma comunidade humana, esta subunidade de conhecimento pode ser reproduzida, modificada, composta com outras e propagada para outros indivíduos da comunidade. Tal conceito foi a inspiração para a definição dos Algoritmos Meméticos no final dos anos 80.

A definição inicial de MAs era uma modificação de algoritmos genéticos, em que seria realizada uma busca local adicional para o problema do caixeiro viajante. Posteriormente, em um arcabouço geral, baseados no mesmo arcabouço de evolução, os MAs utilizam os pequenos pedaços de informação, que definem um comportamento para resolver um determinado problema, no lugar de uma estrutura genética.

Assim, a técnica baseada em memes ganhou notoriedade, pois o comportamento individual é representado por um método de aprendizado aplicado aos indivíduos de um método populacional. Os memes, quando apresentam bons resultados, são copiados e mo-

dificados, gerando novos comportamentos e possivelmente melhores soluções, assim como nos EAs. Nos algoritmos meméticos, o aprendizado ao longo da vida de cada indivíduo e a transmissão desse conhecimento levariam às melhores soluções. A principal diferença desta abordagem em relação à estrutura evolutiva clássica é a realização da busca local após a geração de cada indivíduo.

Um algoritmo memético então pode ser resumido como a composição de diversas estratégias para a busca por soluções ótimas de um problema. A maneira como estas estratégias são combinadas e sua interação gerou uma classificação para os MAs, como detalhado no trabalho de Neri e Cotta [66], que apresenta uma revisão aprofundada da literatura sobre MAs. Nos MAs adaptativos hiper-heurísticos, a coordenação dos memes é feita por regras heurísticas pré-estabelecidas. Nos de aprendizado Meta-Lamarckianos, a ativação dos memes está relacionada ao seu sucesso. Nos MAs autoadaptativos e co-evolucionários, os memes, codificados diretamente nas soluções, ou evoluindo em paralelo a elas, participam do processo evolutivo e sofrem recombinação e seleção. Finalmente, nos adaptativos de diversidade de *fitness*, uma medida de diversidade é utilizada para selecionar e ativar os memes mais apropriados. Outro *survey* relevante sobre este tipo de técnica pose ser encontrado em [15].

Exemplos da combinação entre as técnicas são diversos. A combinação do aprendizado local por meio de memes e técnicas evolutivas por exemplo, pode ser encontrada em [95]. Nesse trabalho, o autor utiliza programação genética com árvores de decisão em um problema de classificação. A partir do treinamento, informações são coletadas para serem utilizadas em uma função de *fitness*. Além disso, uma função de *hill climbing* é introduzida entre os operadores da árvore para realização de aprendizado e busca local.

2.2.2 Meta-Heurísticas Adaptadas

Diante da relevância em se abordar a questão do aprendizado, estudamos trabalhos na literatura que apresentam modificações no conceito original das meta-heurísticas de busca. Apresentamos aqui um pequeno levantamento tendo o PSO como base.

Inspirado por princípios do aprendizado social humano, em [88], é proposta uma modificação no algoritmo do PSO. Nesta modificação, são introduzidas duas estratégias de busca diferenciada para as partículas.

Na primeira estratégia, propõe-se o uso da constante de inércia da melhor partícula de maneira auto regulada. Neste caso, a melhor partícula em uma determinada iteração vai realizar a busca sem interação com outras partículas. Uma vez que deixe de ser a partícula melhor posicionada, o algoritmo retorna para o modo de busca padrão. Esta estratégia faz com que esta partícula possua um caráter mais exploratório. Este conceito é inspirado pelo fato de que, confirmado que uma direção leva ao resultado correto em um processo de busca, um agente segue naquela direção mais rapidamente.

A segunda estratégia é o uso da auto-percepção para escolha da direção de busca. Nesta estratégia, a melhor partícula considera a sua direção como a direção a ser seguida, ignorando a posição coletiva. O conceito é inspirado no fato de quando uma pessoa acredita que está correta, ela tem convicção de sua posição e se preocupa menos com a posição alheia. O restante das partículas leva em consideração o melhor posicionamento

global. O desempenho destas modificações foram avaliadas em 25 funções de *benchmark*, sendo comparada com outras estratégias, apresentando bons resultados.

Em [56], Li et al. apresentaram um mecanismo de troca de informação entre as partículas, e um modelo de regulagem desta troca baseado em cooperação e competição. Para isso, usaram estratégias de diferentes versões do PSO: o PSO clássico como um PSO global, que é propenso a ficar preso em máximos locais tendo em vista a tendência de seguida do G_{best}; o PSO local [50] , em que as partículas levam em consideração o P_{best} de duas partículas em sua vizinhança; e o PSO com o bando completamente informado, no qual todas as partículas de uma vizinhança contribuem com informação de posicionamento.

O modelo de compartilhamento de informação proposto se baseia em uma quadro negro, no qual cada partícula preenche seu P_{best} e o deixa disponível para todas as outras lerem. Conforme as iterações ocorrem, valores antigos são sobrescritos pelos novos valores descobertos. O modelo de cooperação e competição é definido como uma solução para quando uma partícula se encontra imóvel no espaço de busca por um certo número de iterações. Neste caso, a partícula presa em uma posição necessita da cooperação das outras partículas. A escolha de qual partícula irá cooperar vem da competição do melhor P_{best}. O modelo proposto foi avaliado em 16 funções de *benchmark* e comparada com outros modelos de PSO. Os melhores resultados são em funções multimodais com muitos ótimos locais.

2.2.3 Hibridizações de Algoritmos Evolutivos e de Inteligência Coletiva

Após verificarmos a relação de comportamento coletivo e aprendizado individual, nos debruçamos agora em combinações possíveis que fazem com que a informação adquirida coletivamente por uma população se relacione com genes e que genes atuem na configuração de comportamentos.

A utilização em conjunto dos conceitos de inteligência coletiva e algoritmos evolutivos pode ser encontrada em [57]. Neste trabalho, Lovbjerg et al. propuseram duas abordagens. Na primeira, o conceito de reprodução e *crossover* utilizado nos algoritmos genéticos é aplicado em conjunto com o PSO. Após a avaliação do seu posicionamento, cálculo das novas velocidades e deslocamento, ocorre a reprodução entre as partículas. Tal reprodução ocorre probabilisticamente e o posicionamento das partículas-filho ocorrem com o *crossover* aritmético do posicionamento dos pais. A velocidade é definida como a soma normalizada das velocidades dos pais.

Na segunda proposta, subpopulações são utilizadas e, assim, cada subpopulação mantém o seu melhor posicionamento global. Quanto às reproduções, estas podem ocorrer tanto dentro da mesma subpopulação quanto entre subpopulações diferentes. Os filhos substituem os pais para que o número de partículas se mantenha constante.

Seguindo a mesma linha, Higashi et al. [39] incorporaram o conceito de mutação ao PSO, utilizando para isso a mutação gaussiana, muito utilizada em GA. Nesse caso, ao invés de todas as partículas terem seus posicionamentos alterados de acordo com uma probabilidade predeterminada, os indivíduos são escolhidos probabilisticamente e seus

posicionamentos são alterados de acordo com uma distribuição gaussiana, quando do momento da transição de geração do PSO. As partículas do PSO são controladas de acordo com uma expressão matemática que considera as coordenadas da partícula, a componente de velocidade correspondente, a coordenada da melhor partícula e a coordenada do melhor ponto visitado pelo bando.

Similarmente, Poli et al. [73] propuseram que as técnicas de GP sejam utilizadas para encontrar uma expressão que maximize os resultados do PSO ao invés da utilização da expressão clássica ou outras projetadas por humanos. Para isso, durante o treinamento, em cada geração do GP, o PSO é executado por 30 iterações com 10 partículas aleatórias. Repetindo-se este processo 5 vezes, é possível avaliar o resultado obtido das execuções do PSO. Como medida de *fitness*, foi utilizada a distância acumulada nas execuções entre a melhor partícula e máximo global. Em um outro modelo, a distância acumulada entre o máximo global e todas as partículas foi explorada. Para o GP, foram realizados experimentos com 1000 e 5000 indivíduos e o número de geração era interrompido ao atingir o valor de 100 ou até que um dos indivíduos apresentasse um alto valor de *fitness*.

Em [1], os autores propuseram a hibridização das técnicas de ACO, PSO e GA, com a finalidade de clusterização de dados. Quatro abordagens são apresentadas da combinação do ACO e PSO. Nesse método, PSO e ACO compartilham suas soluções e as armazenam em uma tabela de feromônios-partículas.

Na primeira das abordagens, ACO e PSO são combinados de forma sequencial. As melhores soluções produzidas pelo PSO são armazenadas na tabela. Em seguida, o ACO gera novas soluções, utilizando os mesmos valores. Na segunda, ACO e PSO rodam em paralelo e são selecionadas as melhores formigas e partículas ao mesmo tempo. Na terceira, é criada uma tabela de feromônios-partículas estendida em que se dobra o tamanho da tabela. Finalmente na quarta, PSO e ACO rodam separadamente e, ao final de cada geração, é trocado o melhor global de cada abordagem. Finalmente, é proposto um modelo de hibridização que interliga PSO, ACO e GA. Neste modelo, as três abordagens compartilham uma tabela de feromônios-partículas-cromossomos. Inicialmente, o PSO gera as novas partículas e atualiza a tabela. Em seguida, ACO gera novas soluções e substitui as piores soluções da tabela. O mesmo ocorre em seguida com novos cromossomos gerados pelo GA.

No trabalho de Das e Parouha [19], foi construída uma hibridização de PSO e DE em três partes. Após a geração aleatória da população, todos os indivíduos são ordenados por seus respectivos *fitness*. Em seguida, a população é dividida em três partes iguais. Na parte inferior, é aplicado o algoritmo de DE. Na parte intermediária, o PSO é aplicado. Finalmente, na parte superior, é aplicado novamente a DE. As subpopulações então são recombinadas para criar uma nova população. A próxima geração é definida através do elitismo de indivíduos presentes na geração anterior e atual. Repetições de indivíduos são removidas da população e substituídas por indivíduos selecionados arbitrariamente.

Epitropakis et al. [30] constataram que o processo de busca do PSO concentra as melhores posições encontradas até o momento (P_{local}) em torno das soluções ótimas. Se aproveitando desta característica, os autores propuseram uma estratégia de hibridização do PSO com DE. Em cada etapa de evolução do PSO, o conjunto formado pelo P_{local} de cada partícula é evoluída pela DE.

Dessa forma, o processo de busca do PSO se mantém o mesmo, porém sua convergência aumenta rapidamente, uma vez que a DE trabalha sobre o conhecimento social pré-selecionado pelas partículas. Esta técnica se mostrou simples e escalável, possibilitando que diversas variantes fossem implementadas pelos autores com resultados promissores.

2.2.4 Categorização de Híbridos

A partir do estudo das diferentes formas de combinar os EA e as SI em híbridos, se fez necessário estabelecer maneiras de avaliar diferentes combinações. Assim, buscamos na literatura trabalhos que nos fornecessem tais critérios.

No trabalho de Lozano e García-Martinez [58], foi realizada uma revisão bibliográfica de várias técnicas de hibridização. A partir deste levantamento, foram identificadas três linhas de pesquisa para o projeto de híbridos. Para chegar a esta divisão, os autores partiram do princípio de que, de forma geral, as MHs podem ser divididas de acordo com uma maior tendência de intensificação ou diversificação em seu processo de busca.

O processo de intensificação é a capacidade de refinar as soluções, enquanto o de diversificação consiste em explorar o espaço de busca (*exploitation* × *exploration*). Sendo assim, as hibridizações ocorrem de forma a equilibrar os dois conceitos. As categorias de hibridização são definidas como MHs híbridas colaborativas, MHs híbridas integrativas e uma terceira categoria que consiste em substituir componentes de uma MH por um EA de intensificação de diversificação.

São consideradas MHs híbridas colaborativas aquelas em que mais de uma MH está presente, rodando em sequência ou em paralelo, e ocorre uma troca de informação entre as partes. Dentro desta categoria, ainda são realizadas duas subdivisões: trabalho em equipe e retransmissão. No primeiro caso, diversas MHs rodam em paralelo. Em geral, são as que dividem a execução em subpopulações. A troca de informações ocorre por meio da migração de indivíduos das sub-populações. No segundo caso, as MHs são executadas sequencialmente, e cada saída serve de entrada para a próxima execução, como em um *pipeline*. É ressaltado também que a maioria dos híbridos mantém a exploração em primeiro lugar, para depois realizar a intensificação.

Nas MHs integrativas, uma MH se torna uma componente de outra, de uma forma similar a um modelo mestre-subordinado. De forma geral, uma MHs mestre possui a função de diversificação e a subordinada de intensificação, ou vice-versa. Os algoritmos meméticos se encaixam nesta categoria.

Finalmente, no terceiro modelo de combinação, a ideia consiste em substituir um componente de intensificação e diversificação de uma MH clássica por um EA com as duas funções, transformando assim tal MH em uma MH integrativa. Os autores citaram exemplos da literatura para as categorias que foram definidas e além disso, realizaram a proposta de uma técnica de hibridização do terceiro tipo utilizando como base a busca local iterada (ILS).

Em uma revisão de hibridizações de DE e PSO [99], Xin et al. apresentaram também uma categorização das possíveis combinações das estratégias. Para realizar tal análise, os autores ressaltam que uma visão unificada é útil para compreender as características em comum e discordantes de cada algoritmo, para que se possa elaborar uma estratégia

de hibridização. DE e PSO são ambos otimizadores populacionais (PO). Ambos apresentam a mesma estrutura geral de otimizadores populacionais, sendo a principal diferença a amostragem que realizam do espaço de busca. DE realiza amostragem por meio de perturbações diferenciais e *crossover*, enquanto o PSO segue o paradigma de combinar a percepção individual e o aprendizado social.

As categorias definidas pelos autores são a de colaboração, incorporação e assistência. Na classificação colaborativa, otimizadores cooperam na busca no espaço de soluções compartilhando ou trocando informações, mas o modo como realizam a busca é mantido o mesmo. Em contraste, na categoria de incorporação, o modo como a busca é realizada é alterada e cada uma das heurísticas não pode ser separada explicitamente. Nesse caso, é difícil definir a contribuição no ganho do *fitness* pois as MHs estão fundidas em um otimizador único. Finalmente, no modo assistência, a MH caracterizada como assistente não realiza busca no espaço de soluções, mas sim em um espaço diferente que irá ajudar a busca da MH principal, como por exemplo no espaço de parâmetros.

Além desta classificação prévia, também é apresentada uma taxonomia mais profunda. Quatro fatores são levados em consideração. O primeiro fator é o nível de Hibridização. Nesta categoria, é levada em consideração o nível no qual as MHs atuam e em qual são conectadas. Quatro tipos de níveis são definidos: nível de componente, nível individual, nível sub-populacional, e nível populacional.

Operando no nível de componente, um otimizador é utilizado para regular um ou mais componentes de uma solução candidata. Diferentes componentes de um indivíduo podem ser evoluídos por métodos distintos simultaneamente. Operando no nível individual, um otimizador evolui uma solução candidata completa, mas geralmente, não todas as soluções de uma população. Nesse caso, os indivíduos podem otimizar soluções de forma independente, sem a necessidade de aplicar o mesmo método para todos. No nível populacional, o método evolutivo opera simultaneamente em todos os indivíduos da população. No entanto, métodos distintos podem ser aplicados sobre a população em gerações distintas. Como nível intermediário entre o nível populacional e individual, o nível sub-populacional indica uma divisão de toda a população, que são comumente caracterizados como multi-espécies ou multi-bando. Se ambas as MHs trabalham no mesmo nível, ele é caracterizado como uma hibridização homogênea, e por outro lado, hibridização como heterogênea se as MHs são contrastantes.

O segundo fator é a ordem operacional, que diz respeito à sequência temporal da execução das MHs. As categorias definidas são paralela, sequencial e alternada. O terceiro fator é o tipo de transferência de informação. Este se refere à interconexão das MHs, relativas ao fluxo de informação entre eles. As categorias são denominadas unidirecional, quando ocorre troca de informação em apenas um dos sentidos e bidirecional, nos dois sentidos.

O último fator é o tipo de informação transferida. Considerado pelos autores o mais complexo dos fatores em sua categorização, não está relacionado apenas às informações específicas da MH mas também do problema. Tipos comuns de informação transferida são a solução atual, grupo de soluções de elite, melhor solução encontrada até o momento, componentes de solução, parâmetros de controle e informação de *fitness*. Os autores também apresentam diversas outras taxonomias propostas na literatura.

Sendo assim, sob a ótica da composição de meta-heurísticas de EC, o arcabouço proposto EvoCycle busca um equilíbrio entre intensificação e diversificação, através da incorporação dos relacionamentos ecológicos que seus agentes utilizarão para interagirem uns com os outros. Para isso, as populações híbridas utilizadas precisam do aspecto colaborativo de trabalho em equipe, pois múltiplas populações se encontram no espaço de busca e com uma troca de informação multidirecional, pois todos os agentes potencialmente interagem entre si.

Além disso é preciso também uma conexão que torna a hibridização de incorporação. Isto é necessário para uma conexão íntima entre a representação genética e o comportamento coletivo. Por fim, todo o arcabouço executa em um nível sub-populacional pois a pressão seletiva opera em todas as populações presentes, e de forma alternada, no sentido que uma população não segue a execução até o fim para que outra comece sua execução.

2.3 Computação Evolutiva Utilizando Conceitos de Ecologia

Após estudarmos como realizar a conexão necessária entre a camada genética e a de comportamento coletivo do nosso arcabouço, agora investigamos as formas com que as conexões da camada ecológica estabelecem as interações entre as indivíduos das populações na camada coletiva.

Categorizamos então diferentes técnicas de EC que de uma forma ou de outra incluem o comportamento da interação entre indivíduos e meio ambiente ou entre indivíduos que representam espécies diferentes.

Diferentes abordagens serão apresentadas a seguir: a Seção 2.3.1 introduz a questão do uso de mais de uma população enquanto a Seção 2.3.2 trata da questão das variação no número de indivíduos em uma população; na Seção 2.3.3, apresentamos trabalhos que usam o conceito de Vida Artificial para otimização; por fim, a Seção 2.3.4 trata de abordagens que propõem a construção de ecossistemas para uma realizar a otimização a partir do uso de relações ecológicas entre diferentes espécies.

2.3.1 Múltiplas Populações

A análise de relações ecológicas demanda o tratamento de múltiplas populações (de um ou mais algoritmo bio-inspirado). Em se tratando das meta heurísticas de EC, a ideia consiste em fazer com que um número de sub-populações (de um mesmo algoritmo bio-inspirado) cubra regiões diferentes do espaço de busca ao mesmo tempo. Diversas abordagens foram propostas, utilizando variados tipos de MHs populacionais. As sub-populações podem realizar a mesma tarefa ou realizar tarefas distintas, de acordo com as necessidades do problema tratado.

Uma das primeiras abordagens consistiu em utilizar mais de uma população de GAs, geralmente implementadas de forma paralela, configurando um modelo de ilhas [9]. Nesta configuração, partes das populações são trocadas entre as ilhas, simulando um processo de migração. Sendo assim, dois novos parâmetros são adicionados, o tamanho da popu-

lação que migra e a periodicidade da migração. Este modelo em paralelo apresenta bons resultados, pois em geral diferentes sub-populações exploram regiões distintas do espaço de soluções.

Outro exemplo do uso de múltiplas sub-populações é apresentado no trabalho de Yazdani et al. [102]. Nesse trabalho, foi apresentada uma abordagem utilizando o PSO para realizar a otimização em um contexto de ambientes dinâmicos, que possui ótimo variável. As populações de PSO são sub-divididas nas categorias de rastreadoras e buscadoras. As buscadoras são projetadas de forma a encontrar picos de ótimo no espaço de busca, enquanto as rastreadoras realizam um processo de otimização local destes picos e rastreiam os ótimos quando estes se movem. A proposta foi avaliada em mais de 40 funções de *benchmark*, e resultou em melhores resultados na comparação com outros 20 métodos do estado da arte em questão.

2.3.2 Dinâmicas Populacionais

Tendo em vista as flutuações que ocorrem no número de indivíduos em uma população, decorrentes das interações entre espécies diferentes e presente nos processos biológicos, estudamos processos e algoritmos que vêm sendo propostos recentemente, e abordam esse aspecto não tão explorado na computação evolutiva clássica.

O conceito do tamanho da população é um tópico de interesse para a boa resolução em computação natural. Algumas abordagens verificaram o impacto do aumento populacional. Por exemplo em [96], um estudo foi realizado para avaliar a diferença entre aumentar uma população de uma GA única, contra diversas populações menores de GA em ilhas. Para isso, é feito um modelo de população infinita que guia o comportamento das populações GA.

Outra abordagem é a de Zhanshan (Sam) Ma questionou o uso de populações de tamanho fixo nos algoritmos [59]. Nesse trabalho, o autor propôs o direcionamento para uma teoria de modelagem de dinâmica de populações para computação evolutiva a partir da teoria correspondente na Biologia em que são consideradas as variações espaço-temporais do número de indivíduos em uma população.

Dessa forma, ao invés de a população de um algoritmo de EC ter sempre um tamanho fixo, ela irá variar de acordo com um modelo populacional. O problema adotado para a realização de testes é o de encontrar blocos de 1's, com tamanho fixo, em uma sequência de bits. A função de *fitness* é a quantidade de blocos que satisfaz esta restrição.

O primeiro modelo utilizado é o modelo de crescimento populacional logístico, que consiste na solução de uma equação diferencial não linear de primeira ordem. O segundo consiste de uma modelagem estocástica em que o número de indivíduos da população segue uma função de distribuição de probabilidade. Por fim, no terceiro modelo, é utilizada uma outra estratégia proposta pelo autor, denominada Teoria dos Jogos Evolucionária Estendida. Neste modelo, é aplicada aos indivíduos uma função de probabilidade que define se tal indivíduo sobreviverá ou irá morrer em um determinado instante.

Como o tamanho da população é regido por estas funções, não existe a necessidade da aplicação da seleção por *fitness* nos indivíduos, e dessa forma, a população se torna variável, provocando sua dinâmica. Todas as abordagens foram comparadas com a respectiva

aplicação em populações de tamanho fixo e apresentaram resultados superiores.

Um critério diferente, de seleção, baseado no conceito de dinâmica de populações, é apresentado em [78]. Ao invés de selecionar os melhores indivíduos, os piores são selecionados e ocorre uma tentativa de sua melhora a partir do seu reposicionamento em torno das melhores soluções.

2.3.3 Vida Artificial em Problemas de Otimização

Outro ponto a ser considerado é a relação entre indivíduos e meio ambiente. Neste ponto, encontramos esta representação no conceito de Vida Artificial, apresentado no início do capítulo.

Um exemplo deste tipo é o ALife [6], no qual Assad e Packard propuseram um sistema de vida artificial composto por organismos artificiais (Artorgs) que disputam por recursos disponíveis no meio ambiente. Neste mundo artificial (AWorld), ocorre a colonização por meio do processo emergente proveniente da troca e busca de recursos entre os Artorgs.

Baseado no AWorld, foram propostos métodos de otimização baseado em Vida Artificial (ALife) [100]. O algoritmo ALife considera que o Aworld é o espaço de soluções de um problema de otimização, dessa forma cada Artorg é um agente de busca neste espaço e os recursos buscados são os valores da função objetivo. Assim, a colonização de Artorgs ocorrerá no local do espaço que apresenta recursos ótimos. No modelo ALife tradicional, quatro espécies distintas de Artorgs compõem uma cadeia alimentar circular, na qual os dejetos de cada Artorg é o alimento de um Artorg de uma espécie diferente.

Os Artorgs possuem uma energia interna que deve permanecer acima de um limite para que o mesmo continue vivo. O processo de busca e o tempo diminuem sua energia interna e por isso ocorre a colonização nos lugares em que podem suprir a necessidade de energia. Para realizar a busca, os Artorgs seguem um sistema de percepção, baseado na diferença dos valores da função objetivo no ponto em que se encontram e no ponto onde recursos desejados se encontram.

Assim, a partir de uma função de avaliação, um Artorg pode escolher para qual recurso presente em sua vizinhança ele deve se mover. Se nenhum recurso se encontra em sua vizinhança, ele se move aleatoriamente. Após uma certa idade atingida, e com energia interna o suficiente, os Artorgs podem se reproduzir com outros da mesma espécie.

Seguindo o modelo de hibridizações, Yang et al. propuseram um método de otimização baseado em ALife que tem uma conexão com GA [101]. Em seu modelo, as melhores colônias obtidas no ALife servem como população inicial para o GA. Da mesma forma, Gu et al. propuseram um método similar, porém, ao invés de GA, os Artorgs geram uma população inicial para o PSO [36].

Aprofundando o uso de conceitos ecológicos em ALife, Satoh et al. [79] propuseram a inclusão de um predador no processo alimentar, com o objetivo de reduzir o tempo de convergência do algoritmo. O predador pode caçar Artorgs se movendo isoladamente. A razão para isso é que Artorgs que não formam colônias não são promissores. Os predadores funcionam em um nível diferente dos outros Artorgs, formando um ALife em dois níveis. Os predadores também possuem uma energia interna e quando a mesma se encontra abaixo de um limiar, ele realiza a busca por uma presa isolada, o que elevaria sua energia

interna.

Partindo do conceito de que na Natureza uma cadeia alimentar tem em geral no máximo 6 níveis, Yu e Wang [103] apresentaram uma modificação da otimização ALife, definindo três níveis de cadeia alimentar e classificando os Artorgs como superiores, intermediários e inferiores.

2.3.4 Computação com Ecossistemas

Finalmente, buscamos estudar abordagens que propuseram o uso de múltiplos conceitos ecológicos para a construção de um Ecossistema com algoritmos de CN. Apresentamos aqui as abordagens que nos serviram de inspiração para este trabalho tanto na formalização quanto na estrutura conceitual.

Uma das primeiras abordagens nesse sentido foi a de Daniel Hillis [40], no qual foi proposta uma abordagem de co-evolução para problemas de otimização, aplicados especificamente no problema de ordenação de redes. Em sua proposta, duas populações buscam pela solução de ordenação em paralelo, sendo uma população responsável, de hospedeiros, por encontrar redes ordenadas, e uma outra população, considerada de parasitas, buscando por casos de teste para cada rede. As duas populações evoluem no mesmo espaço, sendo os hospedeiros avaliados conforme quão bem as suas redes resolvem os casos de teste. Já os parasitas são avaliados conforme quão bem seus casos de teste encontram falhas nas redes ordenadas. Este modelo de co-evolução impede que as populações fiquem presas em mínimos locais e os testes se tornam mais eficientes.

Vulli et al. [94] apresentaram uma modelagem de soluções de problemas de otimização a partir da criação de ecossistemas artificiais (AES). No entanto, esta abordagem segue um caminho de modelo baseado em indivíduos. Este modelo define uma taxa de mortalidade e nascimentos baseada na densidade populacional no entorno de cada indivíduo, e assemelha-se assim ao modelo logístico para produzir dinâmica populacional.

De forma a ilustrar o conceito, é resolvido o problema de sintetização de texturas binárias. Dada uma textura binária aleatória de entrada, busca-se encontrar os parâmetros ótimos do Campo Aleatório de Markov (MRF) que possa reproduzir a textura de entrada.

Além da dinâmica populacional pelo modelo de indivíduos, os agentes são projetados em um ecossistema artificial predador-presa. Para isso, o ambiente em questão é definido como uma imagem. Os predadores são pontos específicos na imagem e são estáticos. Em cada etapa, eles possuem a capacidade de eliminar um indivíduo que não corresponda com o fundo da imagem, de acordo com uma função de probabilidade. As presas, por sua vez, têm a capacidade de se locomover. A partir de um certo número de etapas vividas, também podem se reproduzir.

Quando ocorre a reprodução, filhos herdam os parâmetros do MRF com uma certa mutação. Dessa forma, ao passar das etapas, sobrevivem as presas que mais se parecem com a textura desejada e não podem ser percebidas pelos predadores.

No trabalho de Pasti et al. [72], foi também feita uma proposta da utilização de conceitos de Ecologia para a construção de uma meta-heurística para resolver problemas de otimização, cujo os autores denominaram inspirada em biogeografia. Uma importante contribuição deste trabalho é a definição de um metamodelo unificado que representa uma

definição abstrata de um ecossistema e seus processos internos.

O metamodelo traduz estes processos em um conjunto de definições matemáticas. O metamodelo tem por definição uma tupla com as características de um ecossistema e um conjunto de operações computacionais. No primeiro elemento da tupla, se encontra a definição de ecossistema como um conjunto de indivíduos em uma representação, o espaço que estes indivíduos ocupam e o conjunto de relações entre esses indivíduos.

Quanto ao conjunto de processos possíveis, e caracterizando a segunda parte da tupla inicial, são abordados os conceitos de dispersão, mudanças ambientais, seleção natural, mutação, fluxo gênico, especiação e seleção de espécies. Em termos práticos, um indivíduo nesse trabalho é definido como um vetor de atributos.

Uma importante caracterização realizada é a definição de uma espécie a partir do conceito de espécie biológica, em que uma espécie é definida como uma população de organismos biologicamente isolados de outros. Em outras palavras, podem se reproduzir apenas entre similares. Através do conceito de centro genético, no qual é definido um conjunto de biodiversidade B e a partir da aplicação de uma função aplicada em todos os indivíduos, mapeiam-se os centros de todas as espécies. Aliada a esta função, é definida a operação de distância genética, para definir se indivíduos pertencem a uma mesma espécie.

Dessa forma, a reprodução ocorre entre indivíduos da mesma espécie, mas gera um terceiro indivíduo que pode ser da mesma espécie dos pais, pode ser de uma espécie diferente dos pais e já pertencente ao conjunto B, ou ainda a uma espécie completamente diferente, incrementando o conjunto B e por consequência a diversidade. Tal variação de indivíduos é obtida por meio da mutação.

O fluxo gênico consiste em retirar indivíduos de uma espécie e colocá-los em meio a outra para transmissão de genes diferentes. Além disso, a competição presente nesse trabalho consiste em escolher um habitat e eliminar todas as espécies menos uma do local, sendo a espécie preservada a que possuir maior *fitness*.

Dessa forma, o algoritmo proposto consiste em realizar a reprodução de indivíduos da mesma espécie; fazer a mutação nos filhos e classificá-los; realizar o fluxo gênico com uma espécie aleatória; fazer a seleção natural e competição por habitat; e repetir estes passos por um número pré-determinado de vezes.

Parpinelli e Lopes [71] aprofundaram a utilização dos conceitos de Ecologia em computação evolutiva ao propor um *framework* denominado ecossistema computacional (Eco-Frame) para otimização. O *framework* é composto por indivíduos espalhados pelo espaço de busca definida por uma função objetivo. Este espaço de busca será o ambiente de vida destas soluções candidatas.

Um conjunto de indivíduos representa uma população e o ecossistema computacional pode conter diversas populações capazes de interagir entre si. Cada população pode ser representada por um algoritmo diferente, contanto que todas as populações consigam lidar com a função objetivo, por exemplo, uma população pode ser representada por um algoritmo genético enquanto outra poderia ser representada por um PSO.

Indivíduos compartilhando uma mesma região do espaço de busca caracterizam um habitat. O espaço de busca pode conter portanto diversos habitats e as populações podem se deslocar pelos habitats, assim, uma população pertence a um determinado habitat em

um determinado instante de tempo. A partir destas definições, duas categorias de relacionamento são definidas: inter-habitats e intra-habitat. Para isso são definidas regras topológicas que controlam quais relacionamentos ocorrem entre indivíduos das populações para relações intra-habitats e quais interações ocorrem entre habitats para relacionamentos inter-habitat.

No caso das relações intra-habitats, podem ocorrer tanto relações positivas quanto negativas, sendo as intra-específicas para indivíduos da mesma espécie e as inter-específicas para indivíduos de espécies diferentes. Para as relações negativas intra-específicas, são consideradas o canibalismo e a competição. Para as relações positivas intra-específicas, são a constituição de sociedades e colônias. As relações positivas inter-específicas são mutualismo, proto cooperação, inquilinismo e comensalismo. As relações negativas intra-específicas são competição, amensalismo, predação, parasitismo e escravidão.

O *framework* foi validado tentando encontrar soluções para a função de Schaffer. Em um primeiro momento, foi utilizado o algoritmo ABC de forma homogênea como populações de indivíduos. Sua implementação funciona da seguinte forma: em primeiro lugar, todas as populações são inicializadas aleatoriamente. Após a inicialização, o *framework* entra no laço de sucessão ecológica. Nesta etapa, todas as populações evoluem seus indivíduos em um número pré-determinado de avaliações. Após cada período de evolução, ocorre a definição de habitats. A definição é feita a partir da escolha de centróides para cada população e um limiar é utilizado a partir da distância Euclidiana entre os centroides, caracterizando cada habitat.

Após a definição de cada habitat, é construída uma matriz de adjacências entre as populações. Após a definição dos habitats, relações intra-habitats ocorrem entre indivíduos. Neste caso, apenas a reprodução é utilizada. Por fim, ocorrem as relações inter-habitas. Neste caso, é aplicada a operação de fluxo gênico. Terminada esta etapa, o laço de sucessão ecológica se repete. Em outros trabalhos, foi feita a variação do número de indivíduos de uma população. Para isso, foi utilizado o modelo logístico de dinâmica de populações, obtendo-se ainda melhores resultados.

Em uma outra aplicação, foram realizados experimentos com populações de algoritmos distintos, caracterizando um modelo heterogêneo. Os algoritmos utilizados foram o ABC e o PSO. Esta abordagem produziu melhores resultados que a abordagem homogênea. Em um outra modificação, a definição de habitats é realizada com clusterização hierárquica e o relacionamento entre indivíduos é realizado probabilisticamente.

Por fim, este *framework* foi validado em uma aplicação complexa de cenário real, que é o problema de predição de estrutura de proteínas, nesse caso, relativas ao modelo AB *off-lattice* 2D. O *framework* foi testado utilizando 4 variações homogêneas, uma heterogênea com todos os algoritmos empregados e ainda outra da mesma forma e variação de população. O modelo heterogêneo apresentou os melhores resultados.

Capítulo 3

Arcabouço EvoCycle

Quando ocorre o processo de evolução, a seleção natural depende não apenas da interação entre indivíduos e o meio ambiente, mas também da interação entre os próprios indivíduos; que podem pertencer tanto a uma mesma espécie quanto a espécies distintas. Em oposição, nos modelos clássicos de EC, o processo de seleção natural é simulado pela seleção de indivíduos com o uso de funções de *fitness*.

Propomos que a seleção de indivíduos de EC tenha menos foco no valor da avaliação de *fitness*, e sim nas interações que seriam consideradas relacionamentos ecológicos no mundo natural, e assim ter um processo evolutivo mais voltado para a evolução sem um propósito específico. Desta forma, temos o objetivo de especificar, implementar e validar um arcabouço de vida artificial que realiza computação evolutiva por meio das relações ecológicas.

Este capítulo apresenta o arcabouço EvoCycle que tem como objetivo permitir a criação de cenários de seleção mais próximos ao da seleção natural. A partir da criação destes cenários, investigamos quais propriedades evolutivas podem ser observadas em simulações. Mais especificamente, buscamos verificar a ocorrência da adaptação com o uso do arcabouço (tratando da questão Q-1) e a validade do processo de busca (tratando da questão Q-2).

O restante deste capítulo é organizado da seguinte forma: a Seção 3.1 trata da especificação do arcabouço EvoCycle; A Seção 3.2 apresenta o algoritmo de evolução associado ao arcabouço; A Seção 3.3 apresenta e discute resultados de simulações voltadas à validação do arcabouço.

3.1 Especificação do arcabouço

O primeiro passo para tratar as questões de pesquisa Q-1 e Q-2 é então a especificação do arcabouço EvoCycle a partir dos conhecimentos adquiridos no levantamento do Capítulo 2. O arcabouço EvoCycle é dividido em três níveis conceituais: genética, comportamento coletivo e relacionamentos ecológicos. A ideia é que na camada genética estejam genes que irão ditar padrões para o comportamento coletivo das populações na camada intermediária. Na camada superior temos as relações ecológicas que ditam como as populações interagem com o meio ambiente e entre si, e os indivíduos da camada co-

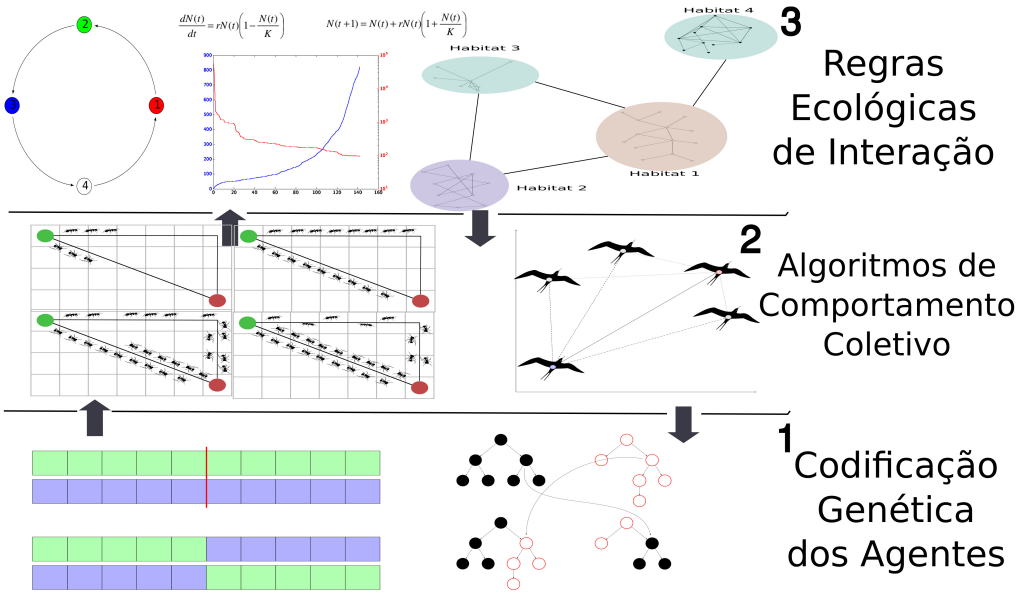


Figura 3.1: Ciclo evolutivo obtido pela integração de diferentes meta-heurísticas posicionadas em diferentes camadas conceituais de organização. Na camada genética, podemos utilizar GA ou GP. Na camada de comportamento coletivo, encontramos por exemplo, PSO. Na camada ecológica, regras de interação entre indivíduos do arcabouço são definidas na forma de relações ecológicas, como predação ou competição por exemplo.

letiva que sobrevivem a essas interações podem transmitir seus genes para descendentes, fechando um ciclo evolutivo. Assim, nosso arcabouço permite a construção de um sistema evolutivo composto de diferentes populações de meta-heurísticas de busca, no qual existem interações entre os indivíduos dessas populações.

A Figura 3.1 apresenta os principais componentes do arcabouço EvoCycle. A camada inferior contempla algoritmos evolutivos, como GP e GA. Como as estruturas destes algoritmos são utilizadas para codificar algoritmos empregados na próxima camada, chamamos esta de camada genética. Na camada intermediária, meta-heurísticas que simulam comportamento coletivo são empregados. Como exemplo dessa classe de algoritmos, temos a classe de SI, como o PSO. Por este motivo, esta camada é chamada de comportamento coletivo. Uma ou mais meta-heurísticas podem ser usadas nessa camada, representando cada uma delas uma espécie diferente. Finalmente, na camada superior, as regras para as interações entre os agentes (da mesma ou de diferentes espécies) são definidas. Estas interações serão as responsáveis por guiar as populações no processo de evolução e são definidas por relações ecológicas. Esta camada é então chamada de relação ecológica.

Comparado aos trabalhos apresentados na revisão bibliográfica do Capítulo 2, uma característica chave do Evocycle é que devido à composição em três níveis, as populações empregadas adquirem a capacidade de apresentar modificações e transmissão de características para descendentes ao longo do tempo, resultando no seu processo de evolução, que ocorre de forma concorrente com a simulação (no sentido de que cada um dos indivíduos pode se reproduzir em momentos distintos, em oposição ao processo de evolução das populações no final de um passo ou até mesmo no fim de uma simulação). Por este motivo, o arcabouço proposto satisfaz o critério de um processo evolutivo sem propósito.

Além disso, uma grande vantagem se encontra na utilização de meta-heurísticas de inteligência coletiva como base para diferentes espécies, fazendo com que os indivíduos da nossa simulação sejam simples, de fácil implementação e modificação. E finalmente, o arcabouço é proposto com uma característica de generalidade, permitindo que diferentes implementações sejam realizadas em instâncias contendo algoritmos que estejam à disposição ou atendam a um requisito específico.

3.2 Algoritmo EvoCycle

O Algoritmo 4 sumariza as principais etapas do processo evolutivo de acordo com o arcabouço EvoCycle. O algoritmo explora características presentes nas correntes da computação evolutiva e tenta integrá-las em uma execução conjunta. Primeiramente, o código genético de cada indivíduo deve ser representado como uma estrutura de EA. Os indivíduos se comportam coletivamente seguindo uma regra de inteligência coletiva (os indivíduos também poderiam realizar aprendizados como o dos Algoritmos Meméticos nesta etapa). Regras ecológicas são aplicadas às populações de forma a controlar o ecossistema. Finalmente, os recursos, tempo de vida, e reprodução de cada indivíduo são os componentes necessários para a construção de sistemas de Vida Artificial.

Algoritmo 4 EvoCycle

```
1: procedimento EvoCycle({p_1, p_2, ..., p_N},R,δ, ET, AT, penal)
2:     inicializar populações POP ← {p_1, p_2, ..., p_N}
3:     T ← 0

4:     repetir
5:         para todo p ⊂ POP faça
6:             Realizar processo de busca utilizando algoritmo da população p.
7:         fim para
8:         para todo i ∈ p faça
9:             alimentar(i).
10:        fim para
11:        para todo i ∈ I faça
12:            se i.Idade > AT então
13:                matar(i).
14:            fim se
15:            se i.Energia < ET então
16:                matar(i).
17:            fim se
18:        fim para
19:        para todo (i, j) ∈ I | dist(i, j) < δ faça
20:            se ∃r(i, j) ∈ R então
21:                aplicar r(i, j)
22:            fim se
23:        fim para
24:        T ← T + 1
25:        para todo i ∈ p faça
26:            aumentar(i.Idade)
27:            penalizar(i.Energia, penal)
28:        fim para
29:    até estar satisfeito
30: fim procedimento
```

No algoritmo EvoCycle, os elementos de busca são modelados utilizando SI, incorporados com EAs e se relacionam por meio de processos ecológicos. $\{p_1, p_2, \ldots, p_N\}$ é o conjunto de populações de híbridos que irão ser empregadas no arcabouço, R é o conjunto de relações ecológicas, δ é a distância para que indivíduos sejam considerados próximos, ET é o limiar de energia para que um indivíduo permaneça na população, AT é o limiar de idade para que um indivíduo permaneça na população, penal é a quantidade de energia que indivíduos perdem ao envelhecer. I é o conjunto de todos os indivíduos de todas as populações e T representa a passagem do tempo.

A execução do algoritmo EvoCycle segue o fluxograma apresentado na Figura 3.3,

funcionando da seguinte forma: o fluxo principal é representado pelas caixas azuis. Primeiramente, populações de MHs são inicializadas com soluções aleatórias, podendo ter inicialmente até N indivíduos em cada população. A inicialização é feita em todas as populações $p \in P$ de agentes de busca i_p que são construídos como híbridos de EAs e SIs (linha 2). As relações ecológicas entre indivíduos estão contidas no parâmetro R. A execução tem início no instante T (linha 3), que é incrementado a cada ciclo (linha 24).

Uma vez que todas as populações são definidas, os atributos de vida artificial são preenchidos com a inicialização dos atributos de idade ($i_p.Idade$) e energia ($i_p.Energia$) de cada indivíduo. A idade codifica quantas iterações um indivíduo tem vivido, enquanto a energia corresponde a quão saudável ele está. Inicialmente, $i_p.Idade$ e $i_p.Energia$ são iguais a zero para todos e são definidos na inicialização das populações.

As populações presentes são constituídas de algoritmos de SI. A cada iteração é executado o processo de busca de cada população, que depende da SI utilizada. Para cada espécie presente no sistema, um fluxo de execução diferente é realizado, representado pelo rótulo 1. O processo padrão de busca se resume em um grupo de agentes movimentando-se pelo espaço de busca seguindo um conjunto de regras pré-estabelecidas e produzindo um comportamento coletivo (linhas 5 e 6). Na nossa solução, essas regras são codificadas em cada agente por meio de uma EA.

Ao terminar uma rodada de busca, os atributos de vida artificial são avaliados como indicado pela seta 2 do fluxograma. Cada indivíduo recebe um acréscimo em sua energia interna proporcional à qualidade do espaço de solução em que se encontra (linhas 8-10), em um exemplo no cenário de otimização, seria obtida por uma avaliação da função objetivo, por exemplo. Neste sentido, a função de alimentar refere-se a aumentar a energia interna de um indivíduo em questão. Nesse caso em proporção ao seu posicionamento.

Após isso, é realizada a avaliação dos indivíduos presentes, verificando se ainda lhes resta tempo de vida e se suas energias internas se encontram em níveis satisfatórios, de acordo com os parâmetros AT e ET (*Age Threshold e Energy Threshold*). No caso negativo, o indivíduo é removido da execução (linhas 11-18). Se dois indivíduos estiverem a menos de uma distância δ no espaço de busca de soluções e existir uma relação ecológica entre os mesmos no conjunto R, a relação existente é aplicada, como indica a seta 3 (linhas 19-23). O processo é repetido até que se atinja uma condição de parada pré-estabelecida na seta 4 (linha 25).

As relações presentes no conjunto R referem-se às diversas relações ecológicas presentes na Natureza. Alguns exemplos imprescindíveis da implementação das mesmas são apresentadas nos Algoritmos 5 e 6, que se referem, respectivamente, às relações de reprodução e predação.

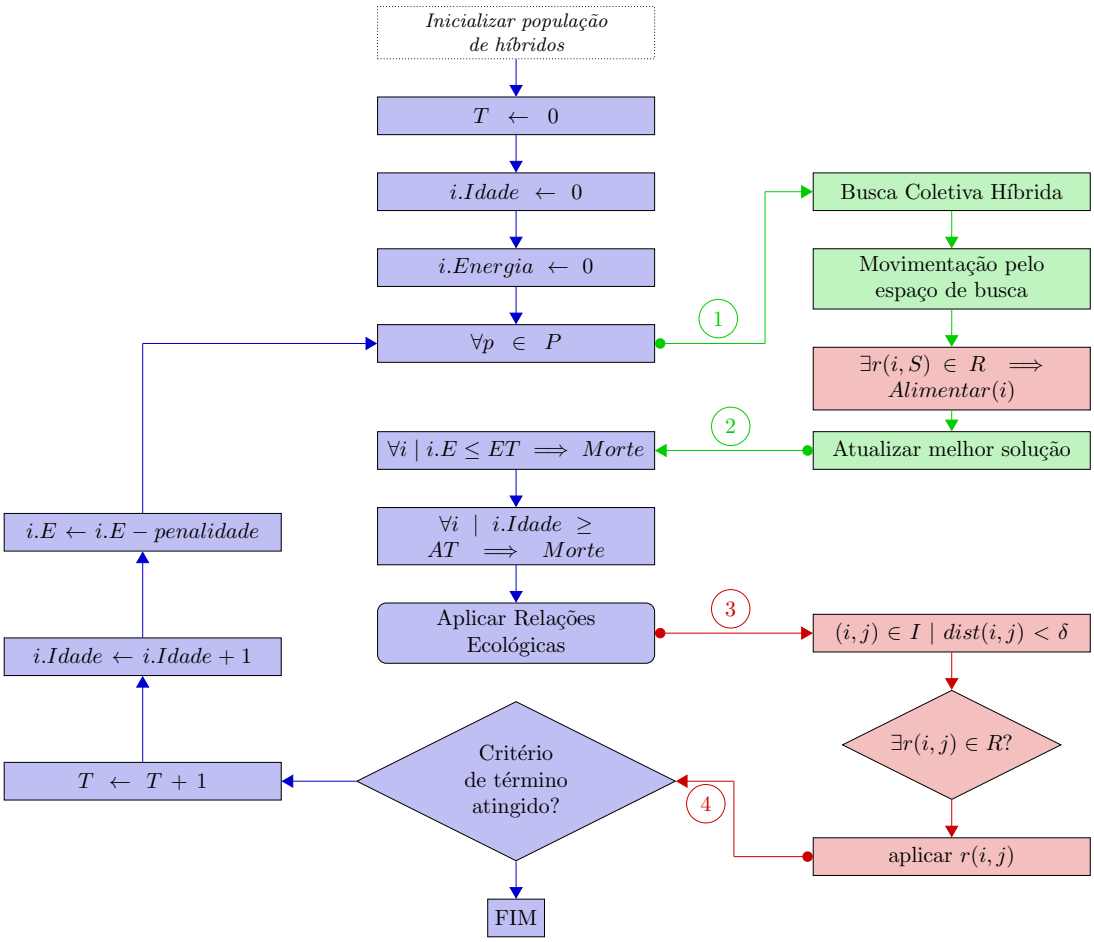


Figura 3.3: Processo evolutivo do arcabouço EvoCycle. O fluxograma começa pela inicialização dos indivíduos de todas as populações. Depois, cada uma delas realiza a sua busca seguindo sua inteligência coletiva. E a cada iteração os indivíduos podem interagir uns com os outros seguindo relações ecológicas e os atributos de vida artificial são atualizados para que se decida quem sobrevive na população.

Algoritmo 5 Reprodução

1: **procedimento** REPRODUÇÃO(i, j, k)
2: k novos indivíduos são gerados com uma combinação do material genético dos indivíduos i e j.
3: **para todo** k **faça**
4: aplicar uma mutação no gene de k, com probabilidade ϵ.
5: **fim para**
6: **fim procedimento**

No Algoritmo 5, apresenta-se um possível pseudo-código para uma chamadas da linha 21 do Algoritmo 4. A relação de reprodução gera novos descendentes a partir do encontro de dois agentes de busca. As operações de EAs são aplicadas para produzir um elemento de SI com um novo código genético.

Algoritmo 6 Predação

```
1: procedimento PREDAÇÃO(i, j)
2:     se fitness(i) < fitness(j) então
3:         matar(i)
4:         alimentar(j)
5:     senão
6:         penalizar(i) e penalizar(j)
7:     fim se
8: fim procedimento
```

No Algoritmo 6, apresenta-se um possível pseudo-código para uma chamadas da linha 21 do Algoritmo 4. a relação de predação permite que um agente de busca de um determinado SI possa aumentar sua energia interna e eliminar outro elemento de outro *Swarm*. A decisão é baseada em uma função de *fitness* de cada SI.

Com essa especificação, garantimos então que o arcabouço atenda aos requisitos apresentados na Tabela 3.1. Fazemos aqui também uma comparação geral com as classes de algoritmos apresentadas no Capítulo 2.

Por utilizarmos diferentes meta-heurísticas na camada de comportamento coletivo, temos múltiplas populações de algoritmos bio-inspirados. Em comparação com os algoritmos clássicos de EA e SI, não são tradicionalmente utilizadas múltiplas populações. No entanto, como existem propostas de modificações com mais populações, consideramos então que esta é uma característica que pode ou não estar presente nessas classes (representadas por um '*'). Já em relação aos simuladores ALife, tradicionalmente não são utilizados algoritmos bio-inspirados como agentes do sistema. Por fim, os sistemas de otimização que propõem uso de relações ecológicas também usam de múltiplas populações.

Como essas populações interagem com as regras definidas na camada de relações ecológicas, apresentamos relacionamentos ecológicos em nosso arcabouço. Nos algoritmos bio-inspirados de EA e SI não são utilizados relacionamentos ecológicos classicamente. Nos simuladores de vida artificial e nos sistemas de otimização com ecologia essa é uma característica presente também.

Os elementos de vida artificial de energia e idade, assim como a relação ecológica de reprodução proporcionam mortes e nascimentos nas populações gerando dinâmicas populacionais. Originalmente, os algoritmos de EA e SI não possuem tal característica, por manterem uma população de tamanho fixo. As simulações de vida artificial buscam incluir essa característica. No caso dos sistemas com ecologia, essa questão também é abordada, embora muitas vezes não decorrente da interação entre indivíduos como propomos, mas gerada seguindo funções matemáticas como a da equação logística para controlar o crescimento populacional.

Os agentes são adaptativos, uma vez que apresentam interação com o meio ambiente na forma do relacionamentos ecológicos de alimentação e entre si, a partir de relações como reprodução e predação. Os agentes respondem a essas interações influenciados pelos genes e pelo comportamento coletivo. No caso dos EAs, não consideramos que possuem esta

Tabela 3.1: Requisitos do arcabouço para integração da Computação Evolutiva e Vida Artificial. O uso de "*" indica que a característica pode ou não estar persente.

Características	EA	SI	ALife	EcologyBased	EvoCycle
Múltiplas populações de EC	*	*	×	✓	✓
Relacionamentos ecológicos	×	×	✓	✓	✓
Dinâmicas populacionais	×	×	✓	*	✓
Agentes adaptativos	×	✓	✓	*	✓
Características hereditárias	✓	×	✓	*	✓

característica, pois as modificações levam a criação de novos indivíduos. Já no caso dos algoritmos de SI, essa é uma característica intrínseca, pois são as modificações dos agentes que geram o comportamento emergente. Também encontramos agentes adaptativos nas simulações ALife, na interação dos agentes com os recursos. No caso dos sistemas com ecologia, essa á uma característica que pode estar presente ou não, dependendo dos algoritmos bio-inspirados utilizados no sistema.

Finalmente, com a construção da hibridização, o uso de algoritmos evolutivos na camada genética possibilita a transmissão de características para descendentes. Aqui temos uma característica intrínseca dos EAs. Já no caso dos de SI, esse fator não é encontrado tradicionalmente, pois não ocorre a criação de novos agentes. Em sistemas de ALife também encontramos essa característica. E mais uma vez, para os sistemas com ecologia a hereditariedade aparece dependendo dos algoritmos bio-inspirados empregados.

3.3 Validação

Considerando-se a especificação do arcabouço, esta seção trata da sua validação. Esta validação é centrada na avaliação das características definidas como requisitos a partir da instanciação de algoritmos de EC propriamente ditos dentro do EvoCyle.

Nesta seção, descrevemos a implementação de uma população usando GP na primeira camada, PSO na segunda camada, e alimentação do meio ambiente e reprodução com as relações ecológicas da terceira camada. Depois avaliamos características da população presente e sua capacidade de busca.

3.3.1 Estrutura Genética

Uma implementação do modelo apresentado na seção anterior será apresentada nesta seção. Aqui usaremos GP na na camada genética, PSO para o comportamento coletivo no contexto de otimização contínua. Ao invés de usar a formulação original do PSO, a função de velocidade de cada agente é definida por uma árvore GP. Similarmente à [73], nós consideramos cada uma das equações provenientes das árvores GP como o código genético, ou genótipo, de cada partícula. Os nós terminais de cada árvores GP são os mesmos elementos das equações PSO:

- X_i é a posição no espaço de busca,

- V_i é a velocidade de cada partícula,
- P_{best_i} é a melhor posição encontrada pela partícula i, e
- G_{best} é a melhor posição encontrada entre todas as partículas.

Para ilustrar como essa árvore se configura vamos utilizar uma equação de velocidade da seguinte forma: $V = \frac{1}{2} \times ((G_{best} - X_i) + (P_{best} - X_i))$. Com essa equação, a árvore correspondente é ilustrada na Figura 3.6.

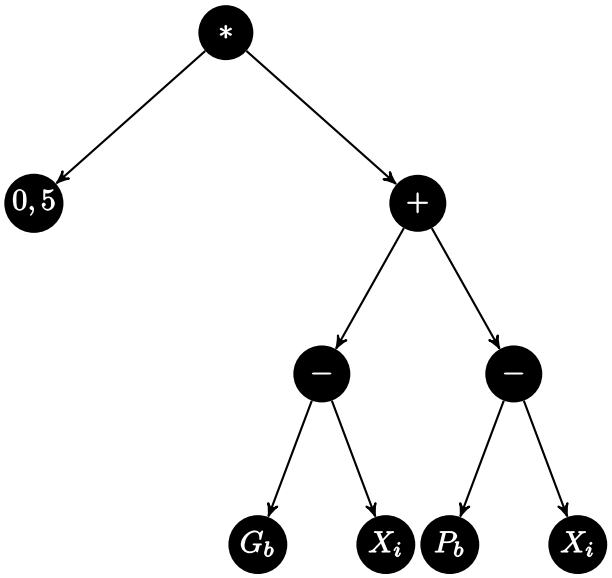


Figura 3.6: Exemplo de uma equação de velocidade representada como uma árvore GP. A árvore representa a equação $V = \frac{1}{2} \times ((G_{best} - X_i) + (P_{best} - X_i))$. Gb e Pb são abreviações para G_{best} e P_{best}.

Após calcular a velocidade, a posição é atualizada por $X_i = X_i + V_i$. A população inicial é gerada aleatoriamente, como apresentado na Seção 2.1.1, referente à criação da população de árvores GP, tendo suas velocidades definidas pelos parâmetros apresentados na Tabela 3.2. Neste caso Pos representa a posição X_i de uma partícula, Vel refere-se à velocidade V_i e $R1$ e $R2$ são números aleatórios entre 0 e 1 obtidos com uma distribuição uniforme. A Tabela 3.3 sumariza os valores iniciais atribuídos aos parâmetros considerados no nosso arcabouço, obtidos empiricamente. As seções seguintes descrevem em detalhes como eles são usados na nossa implementação.

Tabela 3.2: Parâmetros utilizados para geração aleatória de árvores GP.

Parâmetro	Valor
Profundidade Máxima	3
Método	*Ramped*
Nós internos	$+$, $-$, $\times$
Nós folha	$Pos, Vel, P_{best}, G_{best}$
Nós folha	$C1 = 0.5$, $C2 = -0.5$, $C3 = 1.0$, $C4 = -1.0$
Nós folha	$R1$, $R2$ = valores aleatórios entre 0 e 1

Tabela 3.3: Parâmetros utilizados para geração da população inicial de híbridos e para o funcionamento da simulação de vida artificial.

Parâmetros	Valor
Tamanho da população inicial	1000
Tamanho máximo da população	5000
Corte população	80%
Posição	aleatória
Velocidade inicial	0
Idade inicial	0
Energia inicial	0
Limiar de Vida	0
Limiar de idade	100
Distância de reprodução	2
Energia de reprodução	Média
Iterações para reproduzir	10
Penalidade em regiões hostis	-0.001×posição

3.3.2 Modelo de Fluxo Energético

Para construir uma simulação de vida artificial, nós adotamos na simulação uma estratégia em que as partículas possuem perda e ganho de energia relativos à relação com o ambiente, como na alimentação (fluxo energético positivo) e penalização espacial (fluxo energético negativo), como também perda de energia devido ao envelhecimento. De acordo com esse modelo, partículas ganham e perdem energia baseando-se na sua posição no espaço de busca e conforme envelhecem, perdem energia.

Alimentação: o ganho energético por iteração vem da analogia da alimentação, nas quais as regiões do espaço de busca mais próximas do valor objetivo de uma função a ser otimizada serem consideradas mais férteis. O ganho é dado de acordo com a seguinte equação:

$$\Delta E = \frac{10000 \times err_{crit}}{err}, \tag{3.1}$$

em que $err = |f(X_i) - f_{obj}|$ é a diferença entre o valor resultante na posição X_i e o valor objetivo de uma função com domínio real a ser otimizada; e err_{crit} é o valor de erro desejado para parar a execução. Esse ganho de energia foi usado de forma a estabelecer as regiões do espaço de busca de acordo com o valor de erro desejado. Nesta implementação em específico desejávamos ser capazes de encontrar um erro inferior à 10^{-5}, por isso o valor constante na equação. Neste primeiro momento de validação, utilizamos o valor objetivo das funções como uma forma de guiar o processo evolutivo. Em uma analogia ao ambiente natural, isso pode ser visto como se regiões em torno do objetivo fossem mais férteis e dessa forma as partículas poderiam se alimentar de forma mais efetiva.

Envelhecimento: nesta implementação, a cada iteração, as partículas têm a sua idade aumentada por um e perdem energia proporcionalmente a sua idade:

$$L = \frac{Age \times err_{crit}}{Max_{it}} \tag{3.2}$$

em que Age é a idade de cada partícula, err_{crit} é o valor de erro mínimo desejado no processo de busca e Max_{it} é o número máximo de iterações da simulação.

Penalidade: partículas também podem perder energia de acordo com o seu posicionamento no espaço de busca. Quando uma partícula atinge uma região fora dos limites do espaço de busca, a mesma é penalizada da seguinte forma:

$$penal = \sum_{k=1}^{D} 0.001 \times X_k;\ \forall X_k \mid X_k \leq -\delta S \text{ or } X_k \geq \delta S \tag{3.3}$$

em que X_k's são as posições na dimensão D que se encontram fora do limite $\{-\delta S, \delta S\}$, dependendo da função utilizada.

3.3.3 Modelo de Crescimento Populacional

Como as partículas podem se reproduzir e morrer se não possuírem energia o suficiente, o número de indivíduos varia ao longo da execução. Tanto a reprodução quanto a morte de indivíduos depende da quantidade de energia que cada um possui. No começo da execução, para inicializar os níveis de energia, partículas se movimentam pelo espaço de busca por 20 iterações sem perder energia e a reprodução não é permitida. Após isso, a reprodução e a morte de partículas ocorre de acordo com as seguintes regras:

Reprodução: é permitida a reprodução de partículas sempre que suas coordenadas se encontram dentro de um limiar de distância, para todas as dimensões do espaço de busca na qual elas se encontram. Somado a isso, os valores de energia das partículas envolvidas devem ser superiores a um limiar de reprodução. Após uma reprodução, as partículas ficam impossibilitadas de realizar outra reprodução, sendo este fato novamente possível após se movimentarem por 10 iterações.

Mortes: Sempre que uma partícula possui energia menor que um limiar de vida, ela é eliminada da população. Dessa forma, é esperado que partículas sempre obtenham energia para permanecer na população. Para simular a limitação de recursos nas regiões do espaço de busca, uma restrição de 5000 indivíduos foi estabelecida. A motivação para esta escolha é que a população não cresça de forma descontrolada. Sempre que esse limite é atingido, 80% da população são mortos de acordo com quanto de energia cada uma das partículas possui. Os 20% da população com mais energia são mantidos vivos. Partículas com idade maior que 100 também são eliminadas.

3.4 Simulações e Análises

Nesta seção, apresentamos as simulações realizadas referentes às questões Q-2 e Q-3. Inicialmente, verificamos o fenômeno de adaptação com a população híbrida de PSO e GP. Depois fazemos uma avaliação desta simulação no contexto de otimização. Para isso estas questões foram reformuladas como sub-questões específicas, relativas à instanciação do arcabouço apresentada na Seção 3.3.

3.4.1 Questões Específicas

Nossos experimentos buscam abordar as seguintes questões de pesquisa:

QE-3.1 O arcabouço EvoCycle é capaz de produzir frequentemente genótipos de sucesso?

QE-3.2 A aplicação do EvoCycle leva a resultados comparáveis aos dos algoritmos empregados isoladamente no cenário de computação evolutiva?

Para responder essas questões, nós aplicamos o nosso arcabouço em problemas de otimização contínua clássicos, realizando uma simulação de vida artificial com métodos de otimização bio-inspirados como organismos artificiais. Para responder a primeira questão, nós analisamos a diferença da distribuição de genótipos ao longo das iterações da simulação. Além disso, comparamos os resultados de otimização com *baselines* para responder a segunda questão. As seções seguintes descrevem esses passos.

3.4.2 Protocolo de Simulação

O procedimento de simulação consiste em executar o arcabouço buscando no espaço do domínio das funções apresentadas. Cinco funções de teste, sem ruídos foram consideradas. A implementação dessas funções foi baseada na versão 2015 do *framework* COCO para o *benchmark* BBOB.[1] As funções utilizadas são apresentadas na Tabela 3.4.

Elas foram selecionadas por seu uso recorrente na avaliação de MHS, como as realizadas em [71, 73]. Cada uma delas representa uma classe diferente de funções do *benchmark*. A função f1 é uma representante da classe das funções separáveis, a função f8 uma representante das funções moderadas, a função f13 refere-se às funções mal condicionadas, a função f15 representa funções multi modais e a função f21 é uma representante das funções fracamente estruturadas.

As simulações consistiram em executar o arcabouço proposto em 30 instâncias de cada função, fornecidas pelo *framework* COCO. Cada execução foi realizada por 500 iterações. A cada 50 iterações, toda a população era registrada. Para eliminar possíveis redundâncias nas funções de velocidade das partículas, as equações das árvores GP foram simplificadas matematicamente utilizando a biblioteca SymPy[2] antes de serem registradas. Os experimentos foram executados com um limitante de erro de 10^{-7}. Como consequência da variação no tamanho da população, o número total de avaliações de cada função

[1] `http://coco.gforge.inria.fr/doku.php` (Visitado em Junho de 2021).
[2] `http://www.sympy.org/en/index.html` (Visitado em Junho de 2021).

Tabela 3.4: Funções utilizadas na simulação.

Nome	Equação
Esfera (f_1)	$\|z\|^2 + f_{opt}$
Rosenbrock (f_8)	$\sum_{n=1}^{D-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{opt}$
Sharp Ridge (f_{13})	$z_1^2 + 100\sqrt{\sum_{i=2}^{D} z_i^2} + f_{opt}$
Rastrigin (f_{15})	$10\left(D - \sum_{n=1}^{D} cos(2\pi z_i)\right) + \|z\|^2 + f_{opt}$
Gallagher's Gaussiana 101 (f_{21})	$T_{OSZ}(10 - max_{i=1}^{101} w_i exp(-\frac{1}{2D}(x - y_i)^T R^T C_i R(x - y_i)))^2 + f_{pen}(x) + f_{opt}$

Tabela 3.5: Tabela contendo os genótipos da função de velocidade considerando um ou dois literais e um operador. Os símbolos $**$ indicam que o valor está elevado a potência de um inteiro.

X_i	$X_i + G_b$	$X_i - G_b$	$X_i * G_b$	$R - X_i$	$P_b + X_i$	$P_b - X_i$	$P_b * X_i$
G_b	$X_i + P_b$	$X_i - P_b$	$X_i * P_b$	$R - P_b$	$P_b + G_b$	$P_b - G_b$	$P_b * G_b$
P_b	$X_i + V_i$	$X_i - V_i$	$X_i * V_i$	$R - V_i$	$P_b + V_i$	$P_b - V_i$	$P_b * V_i$
V_i	$X_i + R$	$X_i - R$	$X_i * R$	$R - G_b$	$P_b + R$	$P_b - R$	$P_b * R$
$X_i **$	$G_b + X_i$	$G_b - X_i$	$G_b * X_i$	$R * X_i$	$V_i + X_i$	$V_i - X_i$	$V_i * X_i$
$G_b **$	$G_b + P_b$	$G_b - P_b$	$G_b * P_b$	$R * P_b$	$V_i + P_b$	$V_i - P_b$	$V_i * P_b$
$P_b **$	$G_b + V_i$	$G_b - V_i$	$G_b * V_i$	$R * V_i$	$V_i + G_i$	$V_i - G_i$	$V_i * G_i$
$V_i **$	$G_b + R$	$G_b - R$	$G_b * R$	$R * G_b$	$V_i + R$	$V_i - R$	$V_i * R$

também era variável. Uma execução completa, em todas as instâncias de todas as funções, executando em um computador Intel(R)Core(TM) i5-7200U CPU @ 2.50GHz levava em média 8 horas.

3.4.3 Distribuições dos Genótipos ao Longo das Iterações

Nesta seção, investigamos como as estruturas genéticas da população variam ao longo do processo evolutivo, e se existe qualquer tipo de organização predominante. Sendo então o genótipo de cada uma das partículas codificado na equação de velocidade em sua árvore GP, a expressão desse genótipo, ou o fenótipo, seria o movimento resultante causado pela função de velocidade.

Para isso nos baseamos no princípio do equilíbrio de Hardy-Weinberg [29] do campo da genética de populações. Este princípio diz que as frequências genotípicas em uma população devem se manter constantes, de geração para geração na ausência de outras influências evolutivas. Para tanto as condições ideais incluiriam: panmixia, populações infinitamente grandes, ausência de seleção, ausência de mutação e ausência de migrações. Como aqui desejamos aumentar a plausibilidade biológica do nosso sistema, temos justamente o contrário das condições ideais para o equilíbrio, e esperamos verificar uma variação na frequência dos genes, o que por consequência nos mostraria o resultado da simulação da pressão da seleção natural.

Para fazer essa análise, precisamos verificar a variação na frequência dos genótipos ao longo das iterações. Para isso, nós consideramos a combinação de dois literais e um operador como uma medida de direcionamento na função de velocidade das partículas.

Ou seja, os literais da fórmula são um genótipo e o movimento resultante para aquela direção no espaço é o fenótipo associado aquele genótipo. Esta definição vem do fato de que, nas análises realizadas na equação de velocidade do PSO clássico, considera-se uma parte que é direcionada ao melhor posicionamento global e outra ao melhor local. Sendo assim, a fórmula de velocidade de cada agente é analisada como uma combinação de múltiplas unidades possíveis, e com isso somos capazes de observar a frequência dessas unidades dentro de uma população de indivíduos.

Por exemplo, uma partícula evoluída com a equação $V_{i+1} = Gbest \times Vel + Pbest \times R1 \times (R2 + 1.0) + (R1 + 0.5)$ pode ser vista como tendo o seu deslocamento direcionado $1/3$ para a direção $Gbest \times Vel$, $1/3$ para $Pbest \times R1 \times (R2 + 1.0)$ e $1/3$ para $(R1 + 0.5)$. Da mesma forma, poderíamos continuar decompondo a equação, e o termo do meio poderia ser visto como uma composição de metade $Pbest \times R1$ e a outra metade $R2 + 1.0$. Sendo assim, diferentes combinações dos genótipos representam um fenótipo de direcionamento para uma determinada direção. A Tabela 3.5 apresenta o conjunto de combinações genéticas que foram consideradas no nosso estudo.

Dada essa definição dos fenótipos, usando o registro da população ao longo da simulação, as equações de velocidade simplificadas foram avaliadas e foi feita uma contagem dos genótipos para cada instâncias das funções e em todas as dimensões. Esse contagem foi realizada a cada 50 iterações.

Como exemplo dessa contagem, apresentamos histogramas da contagem média de cada fenótipo, feita em 30 instâncias da função esfera em 5 dimensões na Figura 3.7. Nela, apresentamos a contagem de expressões contendo um dos possíveis genótipos, e a variação dessa contagem ao longo de 500 iterações. Estão representados no gráfico os genótipos que predominaram ao longo da execução por meio de histogramas da frequência genética na população. Os genótipos que não predominaram estão agrupados no histograma sob o rótulo "Outros".

Podemos observar que houve um aumento de um pequeno conjunto de fenótipos. No começo da execução, todos os literais estão presentes na população com uma proporção similar entre eles. Nas iterações iniciais, a população começa a ser dominada por fenótipos do tipo $Pbest - Pos$, $Pos + R$, $Pos - R$, $Pbest - Pos$, $Pos \times R$, $Pbest \times R$, e $R - Vel$. Após 150 iterações, a média do tamanho da população é de 3000 partículas, das quais quase 2500 contêm a literal Pos e mais 2000 contêm a literal R. Dentro deste conjunto, mais de 1000 delas são expressões que contêm $Pos + R$ e $Pos - R$, e mais de 500 contêm $Pbest - Pos$. Nos estágios finais da execução, a população passa a ser dominada pelos fenótipos $Gbest - Pos$ e $Gbest - Pbest$. Ao fim, com uma média de 2000 indivíduos, o número de literais $Gbest$ e $Pbest$ é acima de 1500, dos quais quase 1250 são $Gbest - Pos$ que formam esse total. Padrões similares puderam ser observados na execução das outras funções, como pode ser vistos nas Figuras 3.8, 3.9, 3.10, e 3.11.

Também foi computada a correlação da contagem de genótipos e o erro em relação ao valor objetivo de cada função. A Figura 3.12 apresenta alguns exemplos ilustrando a relação da ocorrência de um determinado fenótipo e o erro alcançado no processo de simulação. Os genótipos escolhidos foram: $Gbest - Pos$, por ter se mostrado o mais predominante ao final da execução; $Pos - R$, que apresentou um aumento significativo no meio da execução; e $Pbest - Gbest$, escolhido dentre um dos presentes no rótulo "Outros".

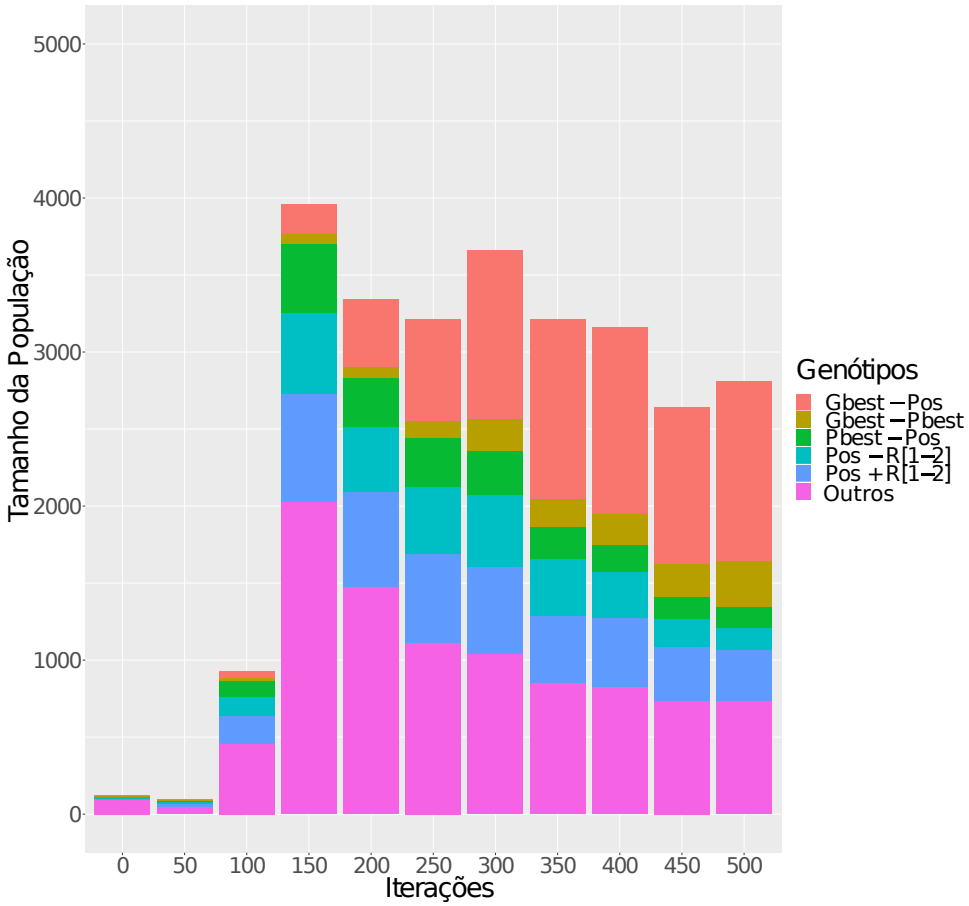


Figura 3.7: Histograma da contagem de genótipos ao longo de 500 iterações na função f1 com 5 dimensões. Cada coluna apresenta os genótipos empilhados a cada 50 iterações. De cima para baixo, o primeiro genótipo é $Gbest - Pos$, seguido por $Gbest - Pbest$, $Pbest - Pos$, $Pos - R$, $Pos + R$, por último a soma da ocorrência de todos os outros genótipos.

Podemos observar que conforme o erro obtido na simulação diminui, o número de genótipos do tipo $Gbest - Pbest$ aumenta. Com o genótipo $Pos - R$ é possível ver que, inicialmente, quando o erro não está tão baixo, sua frequência aumenta. Conforme o erro diminui, esse genótipo diminui, proporcionalmente ao tamanho da população. Esse decréscimo na proporção ocorre devido ao aumento na proporção de indivíduos que possuem o gene $Gbest$. Em adição, existem diversos genótipos que, independente do valor presente de erro na simulação, não apresentam qualquer aumento significativo dentro da população. Por exemplo, a frequência do genótipo $Pbest - Gbest$ não apresenta grande variação na Figura 3.12.

A explicação que apresentamos para estes resultados é de que as partículas foram capazes de se reproduzir uma vez que possuíam energia suficiente. Quando uma região do espaço de busca considerada fértil o suficiente foi encontrada (lembrando que regiões com valores de erro menor, fornecem mais energia), a reprodução das partículas naquela região aumentou. A distribuição dos genótipos nos mostra que, nos estágios iniciais do processo evolutivo, os fenótipos com componentes de busca local foram os que acumularam mais energia e, por consequência, se reproduziram mais. Esse processo de busca de todas as

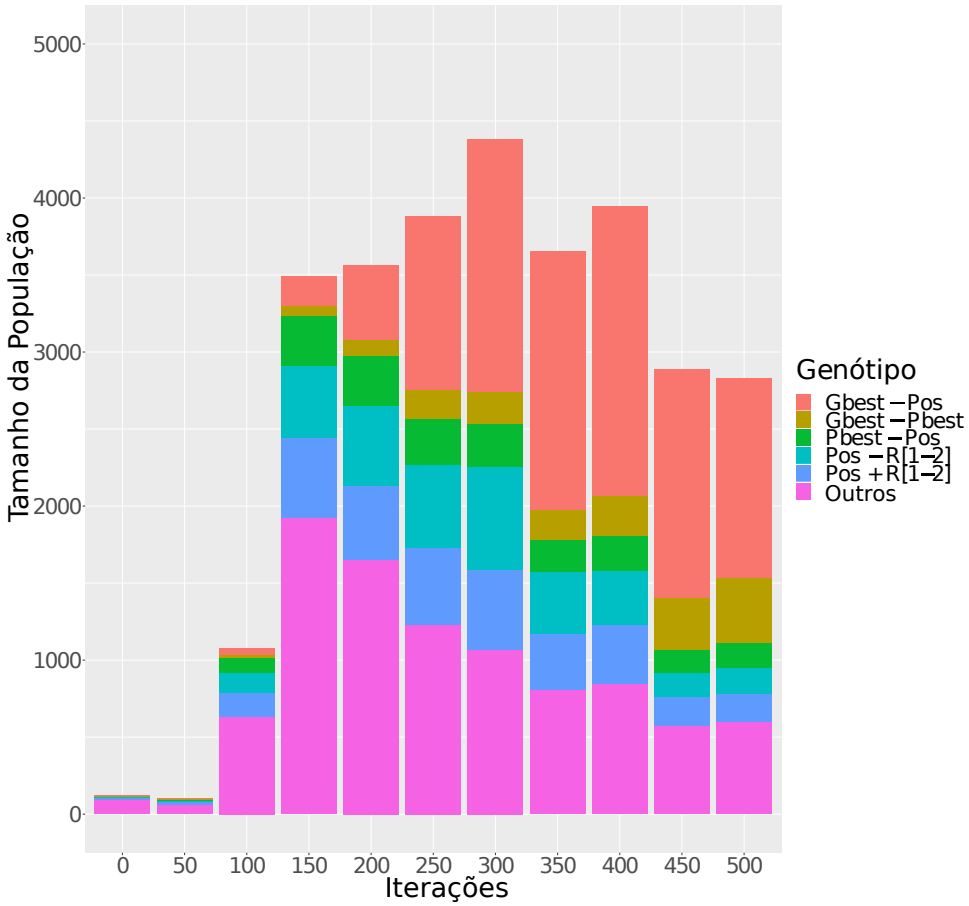


Figura 3.8: Histograma da contagem de genótipos ao longo de 500 iterações na função f8 com 5 dimensões.

partículas como um todo representa uma exploração (*exploration*) do espaço de busca pela população.

Estes resultados mostram que um comportamento global de processo de busca está emergindo. Em um certo ponto, quando regiões com valores de erros ainda mais baixo são encontradas, nós podemos ver que o fenótipo que consistia em se mover em direção a estas regiões e permanecer ali recebendo energia tinham mais sucesso em se reproduzir mais vezes. Nesse caso, podemos ver o domínio deste tipo de fenótipo apresentado nos histogramas, como o genótipo $Gbest - Pos$. Denominamos essa estratégia de busca como *mover-comer-ficar*, que pode ser vista como uma estratégia gulosa. De fato, as regiões mais férteis acabam se tornando mais densamente populadas, em decorrência de um grande número de reproduções. Esse alto número de reproduções pode ser visto como uma busca local no nível populacional, i.e., intensificação (*exploitation*) no espaço de busca, aumentando a chance de encontrar soluções apropriadas no contexto da busca. Dessa forma, concluímos que ao longo do processo evolutivo, certos nichos fenotípicos estão sendo produzidos pelo *framework*. Dessa forma, para resposta da questão QE-3.1, acreditamos que o arcabouço foi capaz de especializar os genes do PSO com a configuração de busca, por meio da adaptação dos indivíduos.

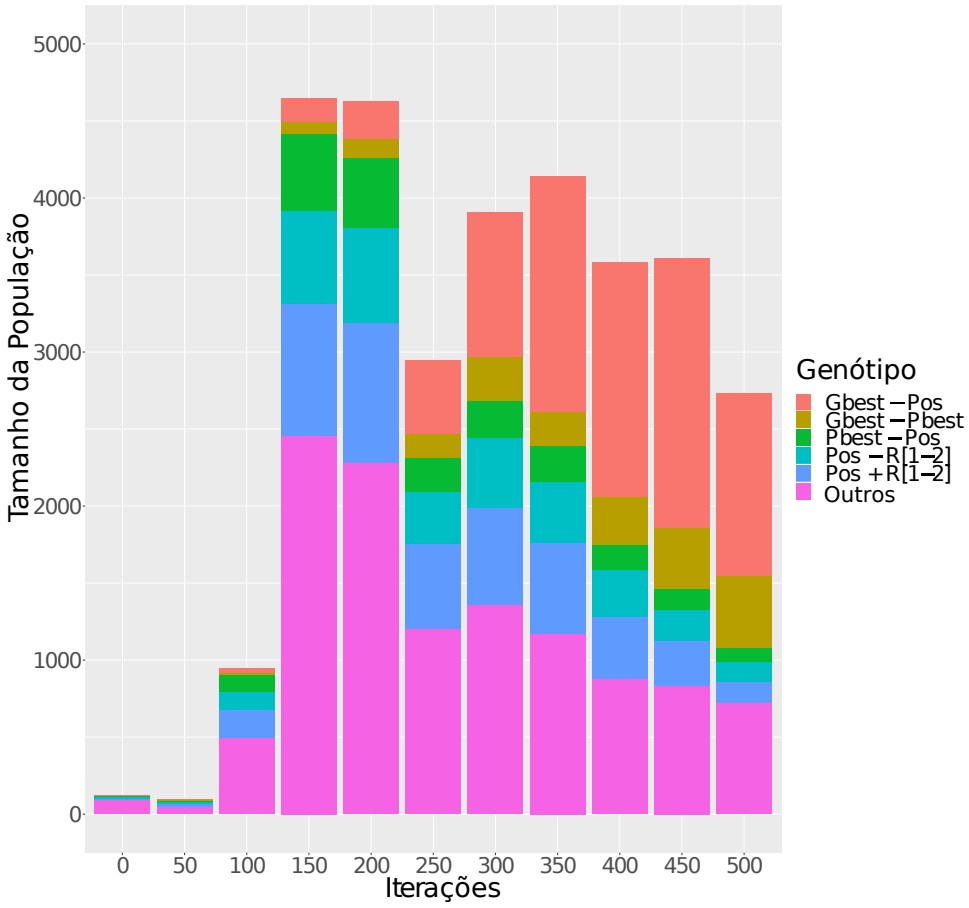


Figura 3.9: Histograma da contagem de genótipos ao longo de 500 iterações na função f13 com 5 dimensões.

3.4.4 Validação da Otimização

Para responder a questão QE-3.2, verificamos também a efetividade do uso do nosso arcabouço em comparação com os algoritmos base isoladamente, no contexto de otimização contínua. Em termos de qualidade de busca, comparamos nossos resultados com o PSO clássico nos *benchmarks* disponíveis do BBOB em [28]. Também apresentamos uma comparação de um PSO que tem suas funções de velocidade evoluídas por uma GP. Para reproduzir os resultados reportados em [73], executamos uma das melhores árvores reportadas referenciadas como PSOG3, com equação: $R_1(Gbest - Pos) - 0.75R_2R_1PosGbest^2 - 0.25R_3R_2R_1PosGbest$.

As Tabelas 3.6 e 3.7 apresentam o número de execuções de sucesso, em 15 tentativas para o PSOGP, PSO e PSOG3, nas funções apresentadas na Tabela 3.4 com 5 dimensões. As tabelas foram geradas com o modulo de pós-processamento da versão 2009 do *framework* COCO para obter os resultados nos mesmos moldes em que [28], com a modificação da avaliação até o erro alvo de 10^{-7}.

Pelos resultados obtidos, é possível observar na Tabela 3.6 que o EvoCycle, usando a população híbrida de PSO com GP atingiu resultados similares ao PSO em 5 dimensões, embora neste caso tenha utilizado mais avaliações da função nos piores casos. Ainda assim, é importante notar que melhores resultados foram obtidos na classe das funções

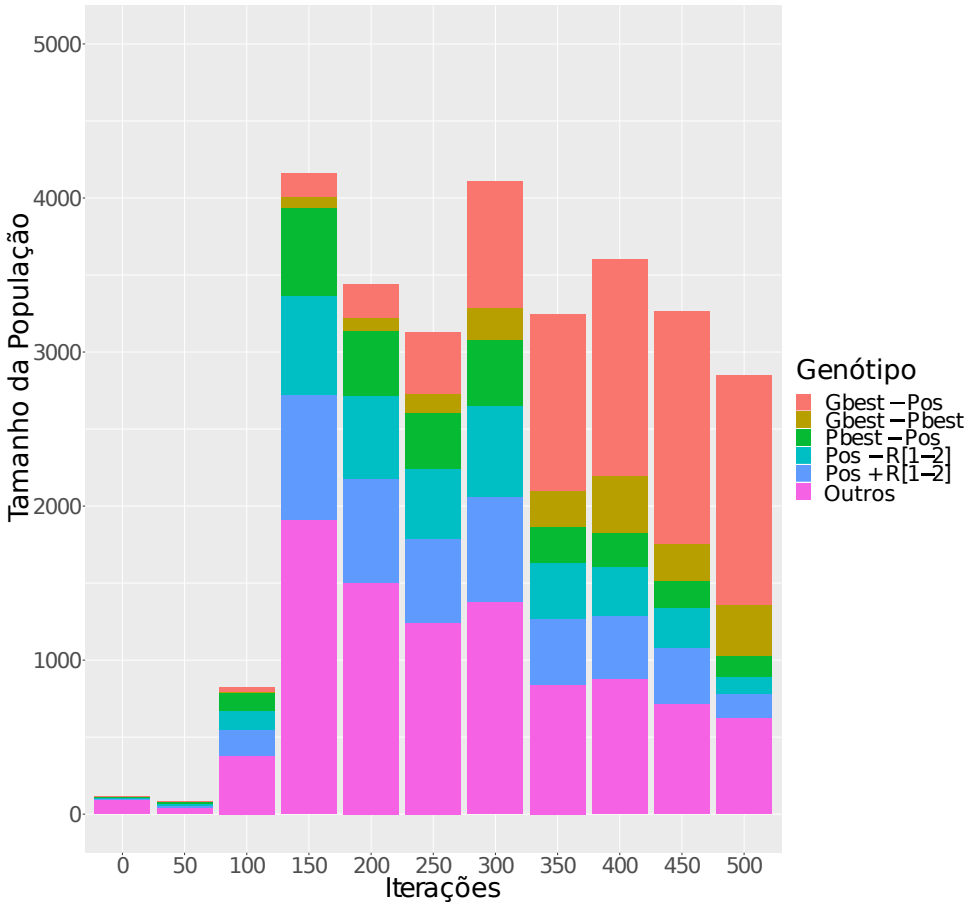


Figura 3.10: Histograma da contagem de genótipos ao longo de 500 iterações na função f15 com 5 dimensões.

mal condicionadas (f13). A Tabela 3.7 apresenta os resultados obtidos na execução do PSOG3. Neste caso, tanto o nosso método como o PSO tradicional apresentaram melhores resultados.

Em ambas as Tabelas, cada célula exibe os resultados referentes a uma das funções. Na linha superior da célula, temos a função correspondente, o número de vezes que foi realizada a simulação nessa função e o número máximo de vezes que a função objetivo foi avaliada. Em seguida, na coluna da esquerda, estão apresentados os valores de erro de minimização. Para cada um dos valores de erro então, são apresentados o número de vezes que a simulação encontrou regiões com erro inferior, o número esperado de avaliações para atingir esse valor, os percentis de 10% e 90% e o número médio de avaliações da função objetivo. Como exemplo tomando a primeira linha da primeira célula, vemos que o PSOGP foi capaz de encontrar um erro inferior a 10 em 15 das 15 tentativas. Para este valor de erro, eram esperadas até 10^2 avaliações da função objetivo. Ao menos uma destas tentativas conseguiu encontrar uma valor de erro menor que 10 com 80 avaliações e 13 das 15 tentativas gastaram no máximo 130 avaliações. Por fim, a média das tentativas de sucesso foi de 100 avaliações da função objetivo.

Sendo assim, ao compararmos o PSOGP com o PSO clássico na Tabela 3.6, notamos que na função f1, ambos obtiveram 15 em 15 sucessos para alcançar regiões de erro inferiores a 10^{-7}. No entanto, na média, o PSO gasta menos avaliações para atingir este

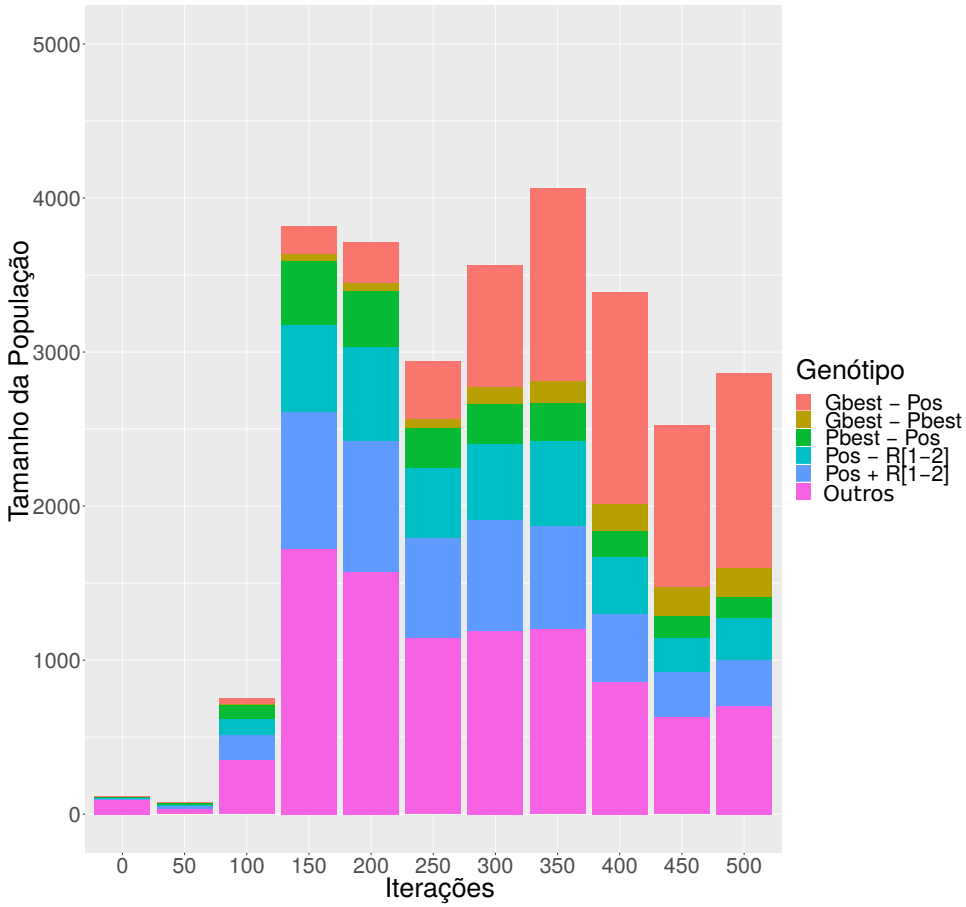
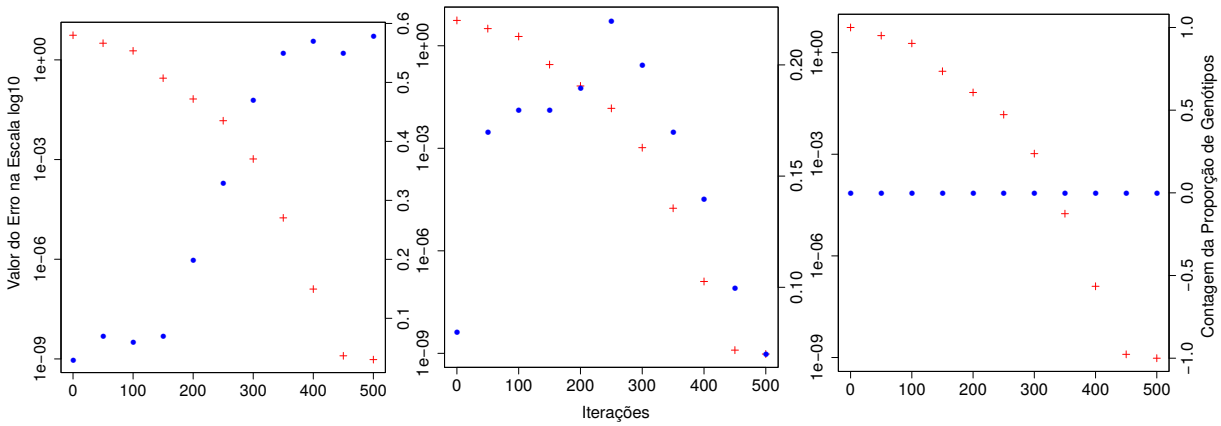


Figura 3.11: Histograma da contagem de genótipos ao longo de 500 iterações na função f21 com 5 dimensões.

objetivo. Similarmente, para a função f8, o PSOGP obteve 13 sucessos em 15 tentativas, enquanto o PSO, 14 em 15. Mais uma vez, observamos que o PSO realiza menos avaliações na média. Para a função f13 o PSOGP foi capaz de encontrar 2 vezes regiões de erro menor que 10^{-5} enquanto o PSO clássico não foi capaz. Por outro lado, para a função f15, o PSOGP não foi capaz de encontrar regiões de erro menores que 10^{-1}. Por fim, para a função f21 o PSOGP teve sucesso de 10 em 15 tentativas, contra 8 em 15 do PSO clássico, porém utilizando mais avaliações na média.

Em relação ao PSOG3, na Tabela 3.7, temos a simulação realizada utilizando o arcabouço tradicional do PSO, porém com as equações de velocidades das partículas evoluídas com GP, segundo apresentado em [73]. De forma geral, observamos que seus resultados não foram capazes de encontrar boas regiões de erro. vale ressaltar que o processo de busca desta implementação foi feito dentro do nosso arcabouço pois os resultados reportados originalmente não possuíam esse formato.

A partir do conjunto destes resultados, verificamos que o PSOGP se compara ao PSO tradicional, no sentido de ser capaz de encontrar regiões do espaço de soluções com valor baixo de erro. Portanto, não afirmamos aqui que possuímos um método melhor de otimização. Por outro lado, estes resultados sugerem que um comportamento de busca foi produzido com sucesso e que a configuração de hibridização entre a camada genética e a camada de inteligência coletiva se mostra funcional.



(a) $Gbest - Pos$ (b) $Pos - R$ (c) $Pbest - Gbest$

Figura 3.12: Gráficos do aumento no número de genótipos e o valor do erro ao longo das iterações. As cruzes vermelhas, com a escala na esquerda, representam o valor de erro na função f1 em 5 dimensões. Os pontos azuis, com a escala na direita, representam o número de partículas contendo um determinado genótipo, em proporção ao tamanho da população. O gráfico da esquerda mostra a correlação do genótipo $Gbest - Pos$. No meio, a evolução da ocorrência do genótipo. $Pos - R$ é apresentado. Na direita, a evolução das ocorrências do genótipo $Pbest - Gbest$. No primeiro caso, podemos observar que o erro diminui conforme o do genótipo aumenta. No caso do meio, o aumento no número do genótipo é visto nas iterações iniciais, seguido pela sua queda nas iterações posteriores. No terceiro exemplo, podemos ver que conforme o erro diminui, não ocorre aumento do genótipo avaliado.

Tabela 3.6: Para uma diferença alvo ao valor objetivo Δf, cada célula da tabela mostra: o número de tentativas de sucesso (#); o número esperado de avaliações para superar $f_{opt} + \Delta f$ (ERT); os percentil 10 e 90 da distribuição *boostrap* do ERT; o número médio de avaliações da função em tentativas de sucesso; o número mediano de avaliações da função para atingir o melhor valor (RT_{succ}). Se $f_{opt} + \Delta f$ nunca foi atingido, letras em itálico denotam o melhor valor atingido Δf da tentativa mediana e os percentis 10 e 90. Além disso, N representa o número de avaliações, e mFE denota o número máximo de avaliações da função executadas em uma tentativa. A primeira coluna corresponde ao PSOGP e a segunda ao PSO. Cada uma das células apresenta os resultados de uma função.

PSOGP

f_1 in 5-D, N=15, mFE=722410

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	1.0e2	8.0e1	1.3e2	1.0e2
1	15	2.1e4	1.2e4	2.9e4	2.1e4
1e−1	15	1.3e5	8.5e4	1.7e5	1.3e5
1e−3	15	3.6e5	3.1e5	4.2e5	3.6e5
1e−5	15	4.2e5	3.7e5	4.8e5	4.2e5
1e−7	15	5.0e5	4.4e5	5.7e5	5.0e5

f_8 in 5-D, N=15, mFE=1.13e6

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	7.5e4	5.3e4	1.0e5	7.5e4
1	14	3.9e5	3.0e5	5.3e5	3.1e5
1e−1	14	5.5e5	4.7e5	6.9e5	4.7e5
1e−3	13	7.7e5	6.1e5	9.6e5	6.1e5
1e−5	13	8.2e5	6.9e5	1.1e6	6.6e5
1e−7	13	8.8e5	7.3e5	1.1e6	7.2e5

f_{13} in 5-D, N=15, mFE=1.19e6

Δf	#	*ERT*	10%	90%	RT_{succ}
10	14	4.6e5	3.6e5	6.4e5	3.8e5
1	11	1.1e6	8.2e5	1.5e6	6.7e5
1e−1	8	1.8e6	1.2e6	2.4e6	8.2e5
1e−3	6	2.6e6	1.7e6	3.9e6	9.2e5
1e−5	2	8.1e6	4.1e6	>2e7	1.0e6
1e−7	0	*69e−3*	*69e−7*	*24e−1*	1.0e6

f_{15} in 5-D, N=15, mFE=1.20e6

Δf	#	*ERT*	10%	90%	RT_{succ}
10	13	3.6e5	2.2e5	5.8e5	2.0e5
1	2	8.0e6	4.4e6	>2e7	8.1e5
1e−1	0	*50e−1*	*99e−2*	*12e+0*	8.9e5
1e−3	.	.	.	.	.
1e−5	.	.	.	.	.
1e−7	.	.	.	.	.

f_{21} in 5-D, N=15, mFE=1.63e6

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	1.4e2	1.0e2	1.7e2	1.4e2
1	12	3.4e5	1.1e5	6.5e5	6.1e4
1e−1	11	5.8e5	2.7e5	9.1e5	1.8e5
1e−3	11	8.0e5	5.1e5	1.4e6	3.9e5
1e−5	10	1.0e6	6.6e5	1.7e6	4.3e5
1e−7	10	1.1e6	7.4e5	1.7e6	4.8e5

PSO

f_1 in 5-D, N=15, mFE=7880

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	4.1e1	3.3e1	4.8e1	4.1e1
1	15	2.7e2	2.5e2	2.9e2	2.7e2
1e−1	15	6.8e2	6.3e2	7.3e2	6.8e2
1e−3	15	2.2e3	2.1e3	2.3e3	2.2e3
1e−5	15	3.9e3	3.7e3	4.0e3	3.9e3
1e−7	15	5.5e3	5.3e3	5.7e3	5.5e3

f_8 in 5-D, N=15, mFE=500000

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	9.6e2	9.0e2	1.1e3	9.6e2
1	14	4.2e4	5.9e3	8.4e4	6.0e3
1e−1	14	6.7e4	3.0e4	1.3e5	3.2e4
1e−3	14	1.8e5	1.4e5	2.7e5	1.5e5
1e−5	14	3.2e5	2.8e5	4.0e5	2.8e5
1e−7	14	4.7e5	4.3e5	5.1e5	4.3e5

f_{13} in 5-D, N=15, mFE=500000

Δf	#	*ERT*	10%	90%	RT_{succ}
10	11	2.1e5	9.0e4	3.5e5	2.7e4
1	3	2.0e6	1.0e6	7.0e6	3.3e3
1e−1	1	7.0e6	3.3e6	>7e6	5.8e3
1e−3	0	*57e−1*	*26e−2*	*21e+0*	2.5e4
1e−5	.	.	.	.	.
1e−7	.	.	.	.	.

f_{15} in 5-D, N=15, mFE=500000

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	8.1e3	2.7e3	1.3e4	8.1e3
1	3	2.1e6	1.1e6	7.1e6	5.6e4
1e−1	1	7.1e6	3.3e6	>7e6	8.8e4
1e−3	1	7.1e6	3.3e6	>7e6	9.5e4
1e−5	1	7.1e6	3.3e6	>7e6	1.0e5
1e−7	1	7.1e6	3.4e6	>7e6	1.1e5

f_{21} in 5-D, N=15, mFE=500000

Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	8.2e1	6.1e1	9.9e1	8.2e1
1	8	4.4e5	1.8e5	7.5e5	1.1e3
1e−1	8	4.4e5	2.2e5	7.5e5	1.6e3
1e−3	8	4.4e5	1.8e5	1.0e6	2.1e3
1e−5	8	4.4e5	1.9e5	7.5e5	3.4e3
1e−7	8	4.4e5	2.5e5	7.5e5	4.5e3

Tabela 3.7: Para uma diferença alvo ao valor objetivo Δf, cada célula da tabela mostra: o número de tentativas de sucesso (#); o número esperado de avaliações para superar $f_{opt} + \Delta f$ (ERT); os percentil 10 e 90 da distribuição *boostrap* do ERT; o número médio de avaliações da função em tentativas de sucesso; o número mediano de avaliações da função para atingir o melhor valor (RT_{succ}). Se $f_{opt} + \Delta f$ nunca foi atingido, letras em itálico denotam o melhor valor atingido Δf da tentativa mediana e os percentis 10 e 90. Além disso, N representa o número de avaliações, e mFE denota o número máximo de avaliações da função executadas em uma tentativa. A primeira coluna corresponde ao PSOGP e a segunda ao PSOG3. Cada uma das células apresenta os resultados de uma função.

PSOGP						PSOG3					
	f_1 in 5-D, N=15, mFE=722410						**f_1 in 5-D**, N=15, mFE=1.00e6				
Δf	#	*ERT*	10%	90%	RT_{succ}	Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	1.0e2	8.0e1	1.3e2	1.0e2	10	15	1.2e2	8.3e1	1.7e2	1.2e2
1	15	2.1e4	1.2e4	2.9e4	2.1e4	1	15	1.5e3	1.1e3	1.9e3	1.5e3
1e−1	15	1.3e5	8.5e4	1.7e5	1.3e5	1e−1	15	3.8e4	2.5e4	5.4e4	3.8e4
1e−3	15	3.6e5	3.1e5	4.2e5	3.6e5	1e−3	0	*11e−3*	*16e−4*	*21e−3*	6.3e5
1e−5	15	4.2e5	3.7e5	4.8e5	4.2e5	1e−5	.	.	.	.	.
1e−7	15	5.0e5	4.4e5	5.7e5	5.0e5	1e−7	.	.	.	.	.
	f_8 in 5-D, N=15, mFE=1.13e6						**f_8 in 5-D**, N=15, mFE=1.00e6				
Δf	#	*ERT*	10%	90%	RT_{succ}	Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	7.5e4	5.3e4	1.0e5	7.5e4	10	15	3.7e3	2.2e3	5.4e3	3.7e3
1	14	3.9e5	3.0e5	5.3e5	3.1e5	1	9	9.2e5	6.0e5	1.8e6	2.5e5
1e−1	14	5.5e5	4.7e5	6.9e5	4.7e5	1e−1	1	1.5e7	7.0e6	>1e7	5.3e5
1e−3	13	7.7e5	6.1e5	9.6e5	6.1e5	1e−3	0	*37e−2*	*11e−2*	*19e−1*	5.6e5
1e−5	13	8.2e5	6.9e5	1.1e6	6.6e5	1e−5	.	.	.	.	.
1e−7	13	8.8e5	7.3e5	1.1e6	7.2e5	1e−7	.	.	.	.	.
	f_{13} in 5-D, N=15, mFE=1.19e6						**f_{13} in 5-D**, N=15, mFE=1.00e6				
Δf	#	*ERT*	10%	90%	RT_{succ}	Δf	#	*ERT*	10%	90%	RT_{succ}
10	14	4.6e5	3.6e5	6.4e5	3.8e5	10	7	1.5e6	8.7e5	2.4e6	3.8e5
1	11	1.1e6	8.2e5	1.5e6	6.7e5	1	0	*11e+0*	*57e−1*	*20e+0*	3.5e5
1e−1	8	1.8e6	1.2e6	2.4e6	8.2e5	1e−1	.	.	.	.	.
1e−3	6	2.6e6	1.7e6	3.9e6	9.2e5	1e−3	.	.	.	.	.
1e−5	2	8.1e6	4.1e6	>2e7	1.0e6	1e−5	.	.	.	.	.
1e−7	0	*69e−3*	*69e−7*	*24e−1*	1.0e6	1e−7	.	.	.	.	.
	f_{15} in 5-D, N=15, mFE=1.20e6						**f_{15} in 5-D**, N=15, mFE=1.00e6				
Δf	#	*ERT*	10%	90%	RT_{succ}	Δf	#	*ERT*	10%	90%	RT_{succ}
10	13	3.6e5	2.2e5	5.8e5	2.0e5	10	14	1.7e5	8.6e4	3.4e5	9.7e4
1	2	8.0e6	4.4e6	>2e7	8.1e5	1	0	*43e−1*	*32e−1*	*86e−1*	5.0e5
1e−1	0	*50e−1*	*99e−2*	*12e+0*	8.9e5	1e−1	.	.	.	.	.
1e−3	.	.	.	.	.	1e−3	.	.	.	.	.
1e−5	.	.	.	.	.	1e−5	.	.	.	.	.
1e−7	.	.	.	.	.	1e−7	.	.	.	.	.
	f_{21} in 5-D, N=15, mFE=1.63e6						**f_{21} in 5-D**, N=15, mFE=1.00e6				
Δf	#	*ERT*	10%	90%	RT_{succ}	Δf	#	*ERT*	10%	90%	RT_{succ}
10	15	1.4e2	1.0e2	1.7e2	1.4e2	10	15	8.5e1	6.2e1	1.1e2	8.5e1
1	12	3.4e5	1.1e5	6.5e5	6.1e4	1	12	4.0e5	2.3e5	6.2e5	1.5e5
1e−1	11	5.8e5	2.7e5	9.1e5	1.8e5	1e−1	12	4.5e5	2.6e5	7.4e5	2.0e5
1e−3	11	8.0e5	5.1e5	1.4e6	3.9e5	1e−3	8	1.1e6	7.1e5	1.8e6	2.4e5
1e−5	10	1.0e6	6.6e5	1.7e6	4.3e5	1e−5	1	1.4e7	6.6e6	>1e7	1.1e5
1e−7	10	1.1e6	7.4e5	1.7e6	4.8e5	1e−7	1	1.4e7	6.8e6	>1e7	2.6e5

Capítulo 4

Efeitos da Predação em Duas Populações

Após a especificação do arcabouço EvoCycle apresentada no capítulo anterior, realizamos uma validação voltada à simulação do processo evolutivo com o surgimento de um comportamento de busca em uma população de híbridos. A hibridização foi construída com a estrutura de árvores GP codificando as funções de velocidade de partículas PSO. Tal combinação permitia que a população evoluísse com o tempo e pudemos então verificar que a influência do ambiente seguindo a alimentação relativa à posição refletia nos genes, favorecendo os genes que levam ao comportamento de busca global.

No entanto, ao realizar essa validação inicial, o foco principal foi verificar a capacidade dessa população evoluir sob influência do ambiente e produzir um comportamento de busca com sucesso. Resta agora validar esses mesmos pontos por completo, mas agora atendendo o requisito de múltiplas populações e verificar a influência de relações entre elas.

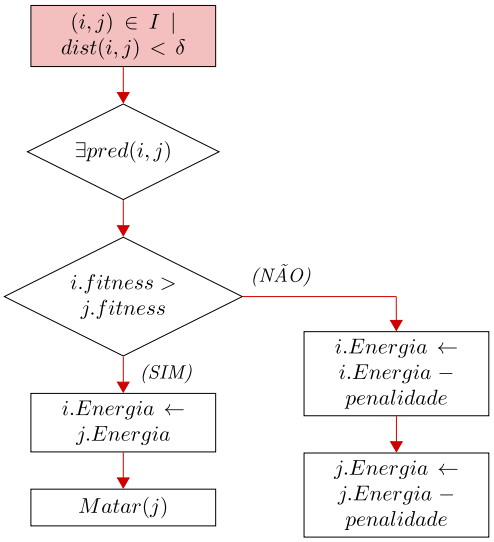
Dessa forma, escolhemos o relacionamento ecológico de predação para verificar se este tipo de interação pode moldar o comportamento da população de híbridos como um comportamento efetivo de busca. Para isso, nos concentraremos agora em realizar simulações do EvoCyle com populações de espécies distintas e com uma relação de predação entre as duas.

Este capítulo é organizado da seguinte forma: a Seção 4.1 apresenta algumas definições associadas ao conceito de predação; a Seção 4.2 apresenta como o relacionamento de predação foi modelado dentro do arcabouço EvoCycle; Resultados e discussões acerca de simulações realizadas são apresentados na Seção 4.3.

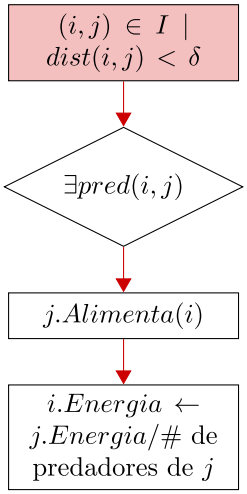
4.1 Predação – Definição

O processo de predação não possui uma única definição na literatura de Ecologia devido às diferentes premissas tomadas ao se tentar definir esse conceito. De acordo com [89], quatro diferentes definições poderiam ser mencionadas, seguindo um nível crescente de generalização:

(a) A primeira e mais intuitiva definição considera que a predação é um relacionamento



(a) Predação de acordo com a definição (a)



(b) Predação de acordo com a definição (b)

Figura 4.1: Fluxograma expandido do relacionamento de predação usando as definições (a) e (b) apresentadas na Seção 4.1.

ecológico entre duas populações de espécies na qual indivíduos de uma espécie, os predadores, se alimentam dos indivíduos da outra espécie, as presas. Como resultado desse processo, indivíduos da população de presas são mortos e eliminados da sua população.

(b) Uma visão mais ampla neste tópico seria considerar a predação como um indivíduo de uma espécie se alimentando de outro indivíduo de outra espécie, mas sem necessariamente matar a presa.

(c) Em outra visão, a predação é definida com uma atividade em que uma população se beneficia de outra de alguma maneira.

(d) Finalmente, como a definição mais generalista, predação é qualquer processo ecológico no qual energia ou matéria é transferida de um indivíduo de uma espécie para um indivíduo de outra.

Adotaremos em nossas simulações a segunda definição de predação, pois esta formulação permite uma implementação mais direta em nosso arcabouço.

4.2 Predation-aware EvoCycle (Pa-EvoCycle)

Neste capítulo, introduzimos o uso de predação no arcabouço EvoCycle. O processo de execução é em geral similar ao apresentado no Algoritmo 4, seguindo o fluxograma da Figura 3.3. No entanto, a parte destacada em vermelho, correspondente aos relacionamentos ecológicos, é detalhada, considerando-se a modelagem do relacionamento de predação, conforme ilustra a Figura 4.1.

Na Figura 4.1a temos o primeiro caso da classificação de predação. Para qualquer par (i_p, j_q), com $p, q \in P$ e uma relação de predação entre eles, tendo o indivíduo i_p predando

o indivíduo j_q, tal relação ocorre com sucesso de acordo com uma condição. Um exemplo seria se o indivíduo i_p possuir mais energia que j_q, então i_p recebe energia de j_q. Tendo a condição para a predação satisfeita, j_q é removido da população. Caso a condição não seja satisfeita, ambos os indivíduos perdem energia.

No caso da predação da definição (b), podemos ver na Figura 4.1b que, caso exista a relação de predação entre os envolvidos, cada presa i_p fornece energia para múltiplos indivíduos j_q. Dessa forma, a energia de i_p é dividida entre todos os indivíduos que os estão predando. Nesse caso, não necessariamente i_p morre ao final do processo.

4.2.1 Questões Específicas

Usando estas definições, temos a hipótese de que o relacionamento de predação pode ser usado para moldar uma adaptação da população, de forma que seus indivíduos se comportem como buscadores efetivos. Em nosso protocolo de avaliação, endereçamos essa hipótese com diferentes perspectivas, listadas a seguir como questões de pesquisa:

QE-4.1 O relacionamento de predação molda o comportamento de predadores? Supomos que predadores que possuem mais energia têm maiores chances de sobreviver e se reproduzir, se forem capazes de encontrar presas com mais energia.

QE-4.2 As estratégias de busca são dependentes dos genes utilizados para codificar possíveis padrões de comportamento. Quais genes são mais significantes no processo de busca? A inserção de genes, possivelmente diferentes dos originais, muda o processo de busca de alguma maneira?

QE-4.3 Cada população possui suas características e usa diferentes estratégias para o processo de busca. Dessa forma, a interação das populações melhora o processo de busca de alguma forma? E como isso pode ser realizado?

4.2.2 Metodologia e implementação

Para responder a questão QE-4.1, o arcabouço foi implementado considerando-se a existência de duas populações diferentes para estabelecer a relação de predação, com o objetivo de criar um sistema ecológico virtual mais diverso. As espécies desse ecossistema de predação utilizam diferentes algoritmos base para sua inteligência coletiva, cada uma representando uma espécie. Sendo assim, a relação de predação é estabelecida a priori da simulação.

A população de predadores é composta pela implementação da hibridização PSO com GP conforme descrita na Seção 3.3, em que a expressão de velocidade da equação do PSO é codificada por uma árvore GP, que representa os seus genes. A população de presas por sua vez é implementada utilizando o algoritmo ABC. A população ABC realiza uma busca por bons locais para construir suas colmeias baseada na qualidade do espaço de busca. As colmeias então acumulam energia, também proporcionalmente à qualidade de sua localidade. Essa colmeias serão a fonte primária de alimento para os indivíduos PSOGP. Para esta implementação, o ABC não foi construído com uma hibridização.

Os indivíduos PSOGP se movimentam pelo espaço de busca, cada um de acordo com uma função de velocidade definida por uma árvore GP. Além disso, dependem de energia para continuarem vivos na população e se reproduzirem, e tal energia pode ser adquirida ao se alimentarem das colmeias ABC. O critério para poderem se alimentar é definido de acordo com sua distância das colmeias, seguindo um limiar pré-definido como um parâmetro do sistema. A ideia é que, após a busca do ABC, os indivíduos PSOGP que foram capazes de se aproximar das colmeias sobrevivam e realizem buscas locais em torno das colmeias.

Para tratar da questão QE-4.2, analisamos a variação da frequência dos genes dos predadores ao longo de diversas iterações. Verificamos o impacto da inclusão de diferentes genes ao introduzir um novo gene na equação de velocidade PSO chamado "Hpos". Este gene codifica a posição de uma colmeia encontrada e atualmente alimenta uma partícula PSOGP.

Uma vez que mais de uma população está se movendo pelo espaço de busca, a informação das melhores localidades encontradas são armazenadas por cada população separadamente. A questão QE-4.3 tem a intenção de avaliar se resultados de busca se beneficiariam do compartilhamento de informação entre as duas populações. Neste caso, comparamos duas implementações, uma em que ocorre o compartilhamento de informação e uma na qual essa troca não ocorre. Essa implementação é feita pelo compartilhamento da melhor posição global encontrada pela população PSOGP e melhor posição de colmeia pela população ABC. Neste ponto, diferente da formulação original do ABC, o melhor posicionamento é atribuído a uma das colmeias.

Resumindo, as simulações realizadas são de populações PSOGP predando populações ABC, em quatro diferentes configurações: com ou sem o gene "Hpos"; e com ou sem compartilhamento de informação acerca da melhor solução global.

4.2.3 Simulações PA-EvoCycle

As simulações do *framework* utilizando predação foram novamente realizadas com o uso de funções de otimização contínua para simular o ambiente como espaço de busca do ecossistema. Ao se aplicar uma função em determinada posição do espaço de busca, é obtido o valor da função e um valor objetivo mínimo. Novamente obtemos um valor de erro para cada processo de simulação, fazendo a diferença do menor valor de função encontrado e o valor objetivo. Nesta etapa, utilizamos as funções sem ruído do *benchmark* BBOB utilizando a implementação 2019 do *framework* COCO [47].

As mesmas funções apresentadas na Tabela 3.4 foram utilizadas: f1 (esfera), f8 (Rosenbrock), f13 (Sharp Ridge), e f21 (Gallagher's Gaussian 101-peaks). Foram realizadas simulações nas 15 primeiras instâncias de cada função, em 5, 10 e 20 dimensões.

4.2.4 Parâmetros

A implementação do EvoCycle é composta de diversos algoritmos funcionando em diferentes camadas. Esta seção apresenta os valores utilizados nos parâmetros em cada algoritmo.

Tabela 4.1: Parâmetros utilizados no arcabouço EvoCycle.

Configuração GP			
Profundidade Máxima	3		
Método	*Ramped*		
Nós internos	$+$, $-$, $\times$		
Nós folha	$Pos, Vel, P_{best}, G_{best}, H_{pos}*$		
Nós folha	0.5, -0.5, 1.0, -1.0		
Nós folha	Valores aleatórios entre 0 e 1		
Configuração PSO		**Configuração ALife**	
Tamanho da população	2000	Limiar de vida	0
Posição	Aleatória	Limiar de distância (L2)	0.5
Velocidade inicial	0	Limiar de reprodução	top 20%
Configuração ABC		Envelhecimento	$\frac{idade}{max_{it}} \times \frac{i}{max_{it}}$
tentativas	100	Iterações para reprodução	10
Número de colmeias	20	Alimentação colmeia (H)	$1/(f(pos)+1)$
ϕ_1 e ϕ_2	Entre 0 e 1	Alimentação predação	$\frac{H}{H_f+d}$

- *Configuração do PSO:* A configuração inicial do PSO utilizada é apresentada na Tabela 4.1. A população inicial é alta para promover uma alta diversidade nos genótipos da população. Todos os indivíduos começam em uma posição aleatória com velocidade inicial zero.

- *Configuração GP*: O código genético utilizado para implementar as equações de velocidade dos híbridos PSOGP, assim como as outras configurações da árvore são apresentadas na Tabela 4.1. Os nós folha são os literais provenientes da equação de velocidade original do PSO. Mais uma vez, Pos é um equivalente para a posição X_i de uma partícula, Vel é equivalente para a velocidade V_i e $R1$ e $R2$ são números aleatórios entre 0 e 1 obtidos com uma distribuição uniforme. O novo gene "Hpos" é utilizado dependendo da configuração experimental. Os nós internos são operadores matemáticos simples.

- *Configuração Alife:* Todas as partículas PSOGP têm um ciclo de vida definidos em termos da quantidade de um valor de energia que possuem. O limiar de vida na Tabela 4.1 apresenta o valor de energia mínimo para que uma partícula sobreviva na simulação. O limiar de reprodução define que apenas 20% dos indivíduos com as maiores quantidade de energia possam se reproduzir. Partículas são consideradas próximas se possuírem uma distância menor que 0.5 em todas as suas dimensões.

- *Configuração de alimentação:* As colmeias de abelha acumulam energia de acordo com a equação:

$$H_{it} = H_{it-1} + E_i \tag{4.1}$$

em que H_{it} é a quantidade de energia acumulada pela colmeia i na iteração it e E_i é a quantidade de energia obtida pela colmeia i na iteração it, seguindo a Equação 2.5.

Os agentes PSOGP realizam a predação e se alimentam das colmeias quando estão próximas o suficiente das mesmas (definidas em termos do limiar de distância). As partículas acumulam energia de acordo com:

$$E_{it} = E_{it-1} + \frac{H}{H_f + d} \tag{4.2}$$

em que E_i é a energia acumulada na iteração it, H é a energia acumulada até a iteração it na colmeia que alimenta a partícula, H_f é o número de partículas se alimentando naquela colmeia, e d é a distância L2 entre a partícula e a colmeia.

4.3 Resultados e Discussão

Após as simulações realizadas, apresentamos os resultados obtidos em termos de otimização na Figura 4.2. Os resultados se referem às simulações nas funções f1, f8, f13 e f21. Nestas figuras, apresentamos o valor de erro (o melhor valor menos o objetivo), obtidos para cada instância, em uma escala log10. O eixo x refere-se às diferentes instâncias, enquanto o eixo y refere-se ao valor do erro.

Da Figura 4.2, notamos que existe uma diferença nos valores de erro das quatro configurações. A primeira observação é que a troca de informação global encontrada por cada uma das populações levou à descoberta de regiões de erro menor em um maior número de instâncias, como mostram as linhas roxa e amarela para todas as funções.

A segunda observação é que, mesmo com as melhoras obtidas, ocorreram grandes flutuações nos valores de erro nas configurações que consideram o uso de compartilhamento de informação do melhor global. Também podemos observar que a introdução do gene "Hpos" gerou resultados similares aos da configuração sem o gene. Esses resultados sugerem que o uso do gene "Hpos" não levou a melhorias (tratando da questão QE-4.2).

Também analisamos a variação do tamanho da população ao longo das iterações em oposição aos valores de erro obtidos na execução. A Figura 4.3 apresenta o crescimento populacional dos predadores PSOGP em dois cenários. O primeiro cenário, sem a troca de informação global entre as populações de predadores e presas, indica que os indivíduos não tiveram tanto sucesso em encontrar colmeias ABC para se alimentar e a população pereceu antes de 100 iterações. Mesmo após a extinção da população de predadores, o algoritmo ABC continua sua execução, porém os erros caíram por poucos valores (na ordem de 10^2).

No cenário com a troca de informação, a população de PSOGP teve mais sucesso em encontrar as colmeias ABC, se alimentar delas, e adquirir energia para se reproduzir. Nesse caso, a população cresceu continuamente. Além disso, a interação entre as populações indica que as posições descobertas pela população de predadores também favoreceram a população de ABC, uma vez que os os erros atingiram valores muito inferiores (na ordem de 10^{-8}). Esses resultados sugerem que a interação entre as populações melhorou a capacidade de busca (questão QE-4.3). Similarmente, as posições encontradas pelo

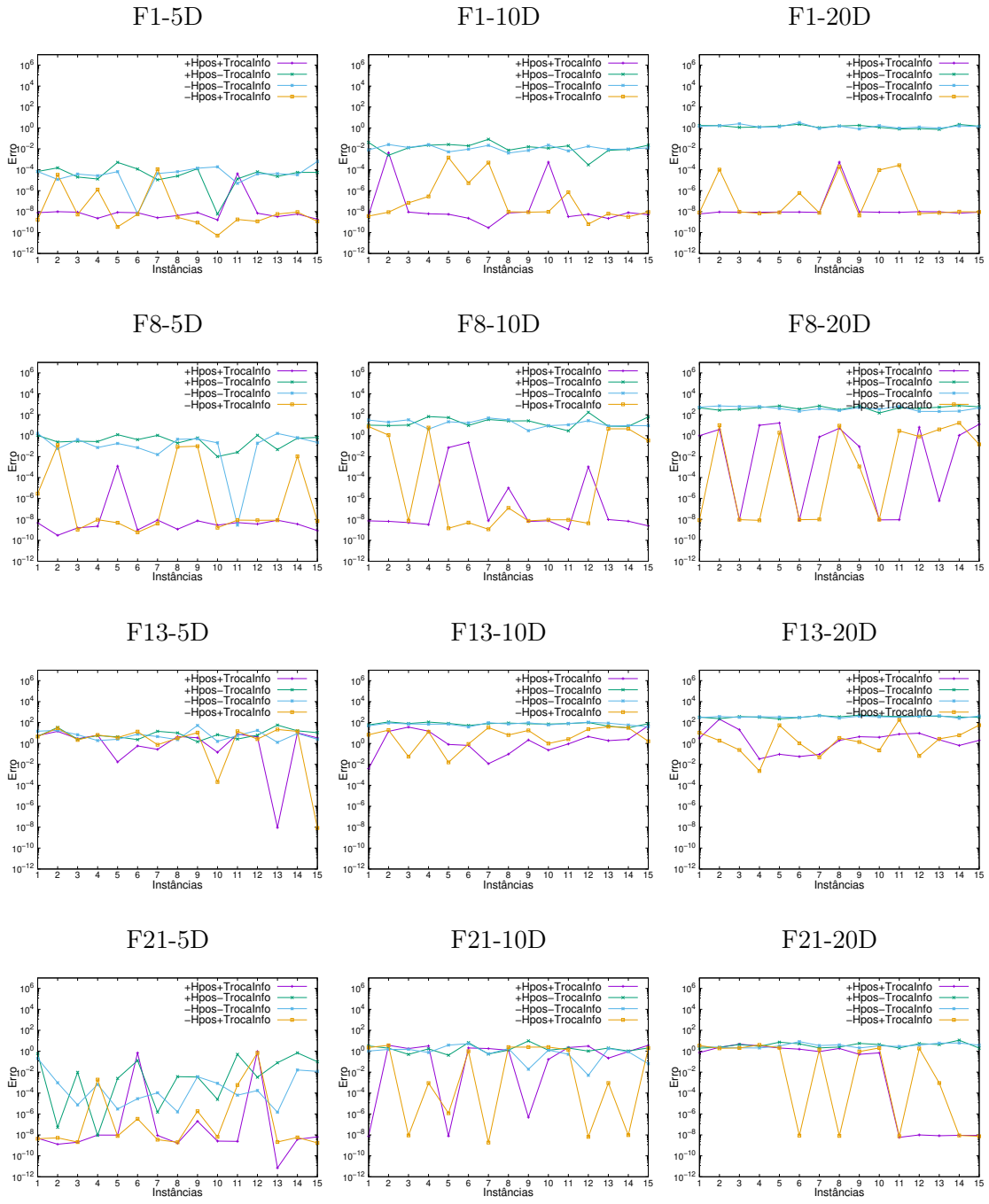
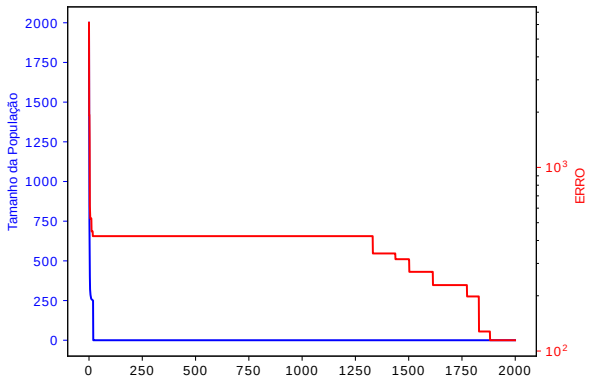
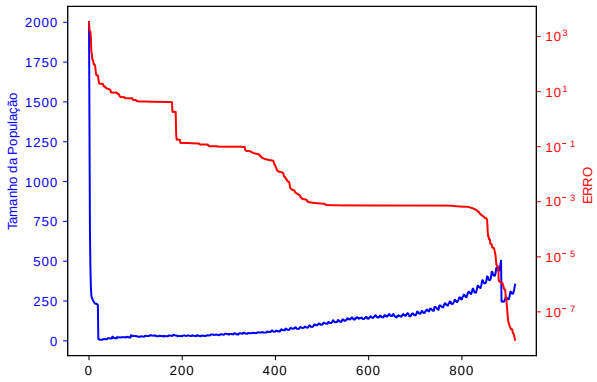


Figura 4.2: Simulações do EvoCycle utilizando diferentes configurações. Cada linha da figura corresponde a uma das funções f1, f8, f13 e f21. As colunas da direita apresentam os resultados em 5 dimensões, na esquerda os de 20 dimensões. No eixo x, temos 15 instâncias de cada um dos problemas. No eixo y, os erros estão representados no intervalo $(10^{-12}, 10^{6})$, em uma escala log10. A linha roxa com marcadores '+' utiliza o gene Hpos e troca de informação entre predadores e presas. A linha verde com marcadores 'x' também utiliza o gene Hpos, mas não faz a troca de informação. A linha azul com marcadores '*' não utiliza o gene Hpos e não faz a troca de informação. Na linha amarela com marcadores quadrados, o gene Hpos está ausente mas ocorre a troca de informação.



Sem troca de informação



Com troca de informação

Figura 4.3: Gráficos do crescimento populacional ao longo das iterações da simulação. A figura de cima ilustra a execução em uma instância de falha, na função f8 com 10 dimensões, e sem troca de informação. Na figura de baixo, um caso de sucesso na mesma função porém com a troca de informação. O eixo x refere-se ao número de iterações. A linha azul mostra o crescimento populacional em número de indivíduos, indicados no lado esquerdo do eixo y. A linha vermelha mostra o comportamento do erro em uma escala log10, indicados no lado direito do eixo y, variando entre 10^2 e 10^{-9}.

ABC favorecem o crescimento populacional de PSOGP. Concluímos então que ambas populações se beneficiam dessa troca.

Para determinar quais genes foram mais relevantes (como formulado na questão QE-4.2), analisamos novamente a mudança na variação das frequências dos genótipos ao longo das iterações, da mesma forma realizada no capítulo anterior. Fizemos essa análise nos casos de sucesso, com e sem o uso do gene "Hpos". Como as partículas de população estão se movendo ao longo do espaço de busca seguindo sua equação de velocidade PSO, diferentes combinações de literais determinam um movimento para uma direção resultante, como um fenótipo. Contamos o número de vezes que diferentes fenótipos aparecem em funções de velocidade de cada indivíduo, a cada iteração.

Ao analisar a contagem de genótipos ao longo da execução, podemos checar quais fenótipos não foram capazes de encontrar colmeias para se alimentar no processo de busca

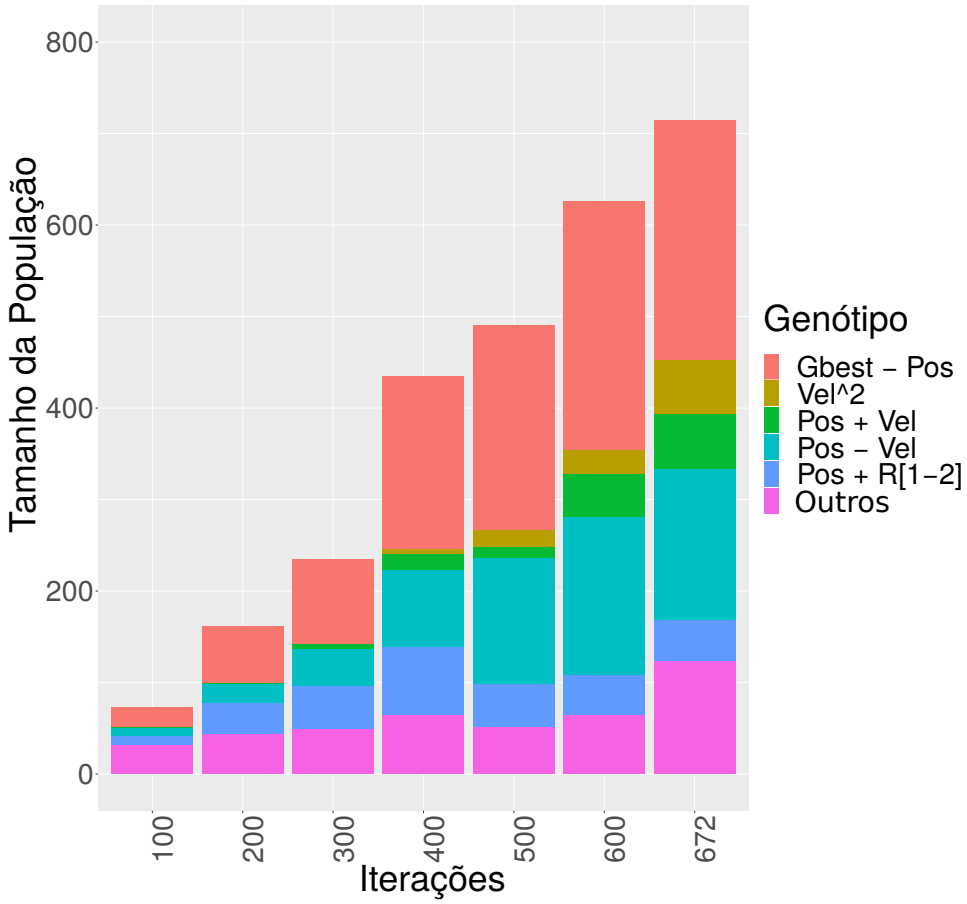


Figura 4.4: Variação de fenótipos por iteração na função f8 **sem** o gene Hpos, com troca de informações de posição. Cada coluna do histograma apresenta o número dos fenótipos mais frequentes. Os fenótipos são representados pelas cores na legenda à direita.

e quais foram. Dentre os fenótipos que permaneceram ao final da execução, podemos verificar se houve algum que prevaleceu sobre os outros na população, baseado na contagem dos genes desses fenótipos remanescentes. Selecionamos então os mais frequentes para analisar.

As Figuras 4.4 e 4.5 apresentam a variação da frequência dos fenótipos em uma simulação do *framework* na função f8 com 10 dimensões. As frequências são apresentadas em histogramas, empilhadas a cada 100 iterações. Cada coluna do histograma apresenta o número total de indivíduos na população a partir da iteração 100 até o fim da execução. Cada coluna é então dividida, com cada parte sendo o número de indivíduos que contém um determinado fenótipo.

Das Figuras 4.4 e 4.5, podemos observar que houve genes mais importantes quando o processo de predação ocorria. Em ambos os casos (com e sem o gene "Hpos"), observamos que se mover em direção à melhor localização global já encontrada ($Gbest - Pos$) domina a população. Esse resultado sugere que a predação foi capaz de produzir um comportamento específico na população, como questionado em QE-4.1. Com os fenótipos dominantes, também podemos observar a emergência de diferentes comportamentos de movimentação. Como observado na Figura 4.4, houve a dominância de cinco genótipos diferentes, associados a quatro fenótipos. O primeiro fenótipo é o mencionado ($Gbest - Pos$),

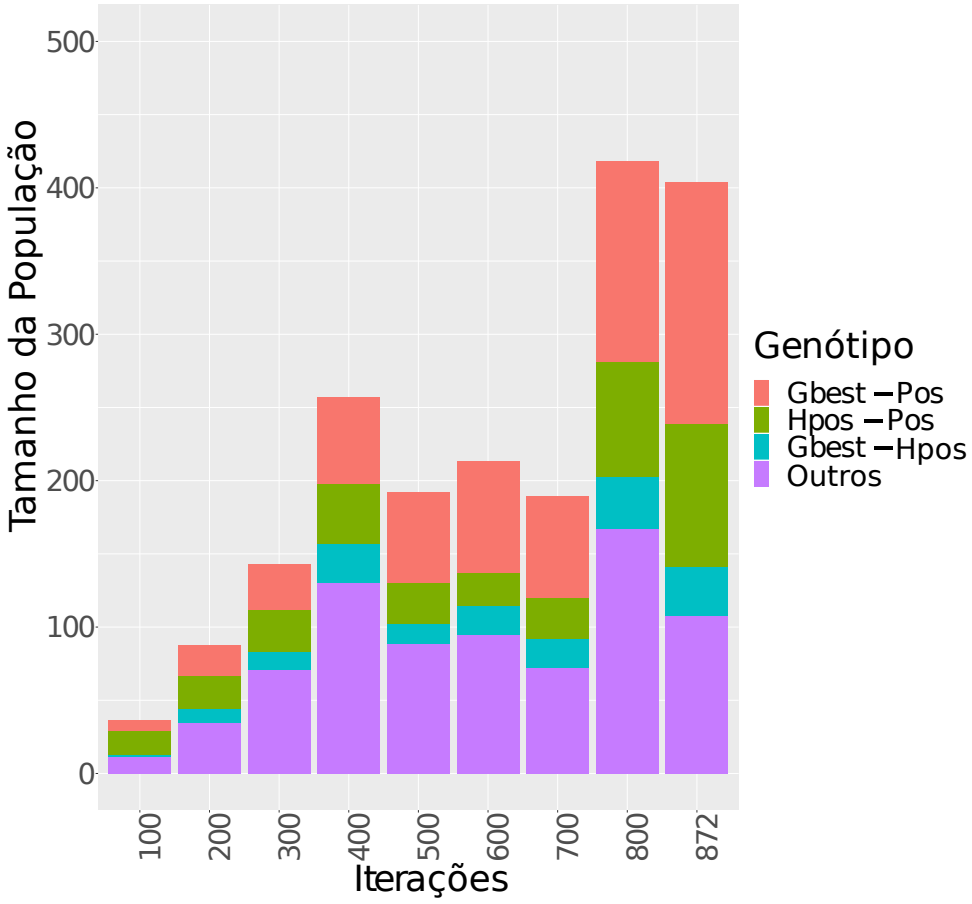


Figura 4.5: Variação de fenótipos por iteração na função f8 **com** o gene Hpos, com troca de informações de posição. As legendas são as mesmas da Figura 4.4.

que se refere ao movimento em direção ao melhor global. O segundo fenótipo faz as partículas continuar o caminho que já estavam seguindo. Esse fenótipo é associado a dois genótipos: $(Pos + Vel)$ e (Vel^2). O terceiro é um fenótipo que leva as partículas a se mover na direção oposta a que estavam vindo $(Pos - Vel)$. O último é um fenótipo de movimento em torno da posição atual, o de busca local $(Pos - R)$.

Já no caso do uso do gene "Hpos", apresentado na Figura 4.5, além da predominância do genótipo $(Gbest - Pos)$, podemos observar um movimento em direção à colmeia que a partícula estava se alimentando, $(Hpos - Pos)$. Além disso, também ocorre a exploração na direção entre a melhor posição conhecida e a posiça da colmeia que o alimenta $(Gbest - Hpos)$.

Capítulo 5

Resiliência Ecológica no EvoCycle

Considerando o sistema ecológico estabelecido nos capítulos anteriores, diferentes análises similares a estudos biológicos poderiam ser realizadas. Neste capítulo, exploramos conceitos inovadores relacionados ao estudo de resiliência em sistemas complexos utilizando a instanciação do arcabouço EvoCycle com as populações de PSOGP e ABC e a relação de predação entre elas.

Como vimos no Capítulo 4, notamos que existe uma conexão entre a dinâmica populacional do PSOGP e a capacidade das populações PSOGP e ABC encontrarem regiões férteis (baixo erro) no espaço de busca. Esta conexão se refletiu em um cenário no qual ocorre a extinção da população de PSOGP em regiões inférteis e outro no qual a população de PSOGP cresce progressivamente. Por este motivo, buscamos uma maneira de avaliar quão estável seria a população de PSOGP. Acreditamos que a análise de resiliência ecológica é uma forma promissora para realizar essa avaliação.

Sendo assim, o objetivo do estudo descrito neste capítulo é investigar informações acerca das dinâmicas produzidas na população de predadores que possam ser utilizadas tanto para a criação de populações resilientes em ambientes de vida artificial, quanto para a melhoria nos processos de busca de computação evolutiva.

Este capítulo é estruturado da seguinte forma: começamos com uma definição do conceito de resiliência na Seção 5.1. Em seguida, apresentamos na Seção 5.2 como incluímos este conceito no arcabouço EvoCycle. A configuração dos experimentos realizados é feita na Seção 5.3 e discutimos os resultados na Seção 5.4.

5.1 Resiliência – Definição

De maneira geral, o conceito de resiliência define o quão estável um sistema complexo se encontra em um determinado momento. Este conceito é utilizado em várias áreas do conhecimento de diferentes formas. Exemplos proeminentes que o utilizam incluem a resiliência ecológica, como a análise da mudança climática global [55], no qual verifica-se a capacidade do sistema climático manter o clima como o conhecemos, a resiliência de ecossistemas, ao se verificar a capacidade de recuperação de um ecossistema após sua degradação ou interferências externas, como a transição entre tipos de vegetação em determinadas regiões (savana × cerrado) [42] e a morte de ecossistemas [82]. Outro

exemplo refere-se ao uso da análise de resiliência nos campos de engenharia e tecnologia. Em engenharia de materiais, por exemplo, análises de resiliência são exploradas para aferir a capacidade de um material receber deformações elásticas e retornar ao seu estado original [37], ou ainda a capacidade de uma rede manter o funcionamento quando ocorrem falhas nos seus componentes [74]. Também encontram-se trabalhos voltados à análise de resiliência na área de Ciências Sociais. Exemplos incluem o estudo da resiliência psicológica, a capacidade de um indivíduo se adaptar diante de fatores de estresse [80]; ou da resiliência organizacional, que avalia a capacidade de um empreendimento se adaptar ao mercado [7]. Dada a grande aplicabilidade deste conceito, diferentes formas foram propostas para que fossem obtidas medidas de resiliência em cada um de seus campos [81].

Como propomos um arcabouço de vida artificial na forma de um ecossistema, neste trabalho nos interessamos especificamente pela resiliência ecológica aplicada a ecossistemas. Diferentes técnicas também foram propostas para quantificar essa resiliência, obtendo um indicador de quando um ecossistema se encontra em risco. Aqui nos concentraremos em uma categoria denominada *critical slowing down*, que tem como princípio avaliar a queda do número de indivíduos de uma população e verificar se essa população se aproxima de um ponto crítico do qual dificilmente seria possível se recuperar. Dentro desta categoria, escolhemos o indicador da taxa de recuperação para ser aplicado no EvoCycle.

A ideia de avaliar a resiliência de ecossistemas por meio da avaliação da taxa de recuperação passa por diferentes etapas, que servem de base para muitas das técnicas de avaliação de resiliência. A primeira delas é a definição de estados distintos dentro dos sistemas. Em seguida, precisamos definir uma característica de estresse do sistema que tem influência sobre a mudança entre os estados. Dessa forma, conforme a variável de estresse considerada aumenta ou diminui, ocorrem transformações no sistema. Por fim, há a etapa dedicada à definição do quão fácil ocorrem estas mudanças de estado. Caso pequenas mudanças na variável de estresse provoquem mudanças de estado, então este sistema é considerado pouco resiliente. Por outro lado, se grandes mudanças são necessárias na variante de estresse para que uma mudança de estado ocorra, então este é considerado um sistema resiliente em relação a tal variável.

Esta definição é denominada de estados estáveis alternativos e a forma como ocorrem as mudanças dos estados também é outra questão de interesse nos estudos de resiliência. A maneira mais tradicional de transição é na qual existe uma correspondência entre a variável de estresse e o estado do sistema, quase que de forma direta. Outra forma importante nesse contexto é a de situação de histerese. Neste cenário, existe também uma correspondência entre a variável de estresse e os estados do sistema, mas acredita-se que existem pontos em que para os mesmos valores da variável de estresse, diferentes estados do sistema podem existir, dependendo do valor anterior da variável de estresse.

Os pontos nos quais as mudanças de estados ocorrem, chamados de *tipping points*, também vêm sendo explorados em vários estudos [42, 93, 91]. Estes estudos se concentram no fato de que dentre os sistemas complexos abordados, existem estados que são de maior interesse que outros (um clima favorável à presença de vida contra um clima no qual não é possível existir vida, no cenário de mudança climática, por exemplo). Dessa forma, a identificação dos *tipping points* se torna relevante pelo fato de se tornar uma forma de tentar prever uma mudança no sistema de interesse.

Para ilustrar estes conceitos, vejamos o seguinte exemplo: em um sistema vegetal seria possível medir a quantidade da população pela biomassa de todos os indivíduos. Neste exemplo hipotético, a quantidade de biomassa estaria relacionado à quantidade de fósforo disponível no solo. Dentre os possíveis cenários, um deles é o que existe uma correspondência direta entre a biomassa e o fósforo do solo. Nesse caso conforme aumenta disponibilidade do fósforo aumenta também a biomassa, conforme ilustrado na Figura 5.1.

Outra situação, a de histerese, é a de quando existe uma relação sobreposta dos estados estáveis alternativos entre a biomassa e o fósforo. Nesse caso, conforme o fósforo se torna menos disponível, a biomassa diminui. No entanto, existe um ponto no qual caso o fósforo diminua a um certo patamar, a biomassa passa a um outro estado muito diferente do anterior. Neste caso, ainda que houvesse recuperação do fósforo, a biomassa não retornaria ao estado no qual se encontrava diretamente. A esta situação é dado o nome de estados estáveis alternativos e a curva que define é chamada de histerese, ilustrada na Figura 5.2.

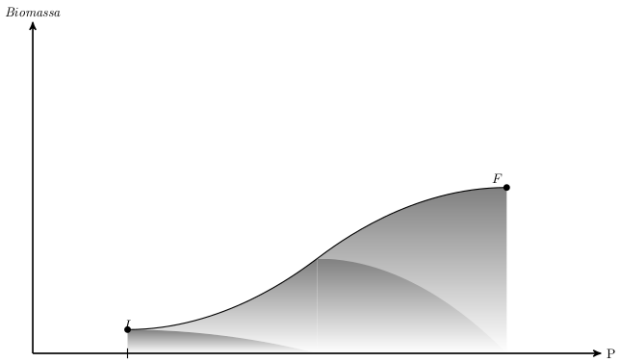


Figura 5.1: Estados estáveis alternativos no qual o crescimento de massa é proporcional a um recurso. Neste exemplo, temos uma medida de Biomassa em função da quantidade de fósforo (P) disponível ao sistema, do instante inicial I até o final F.

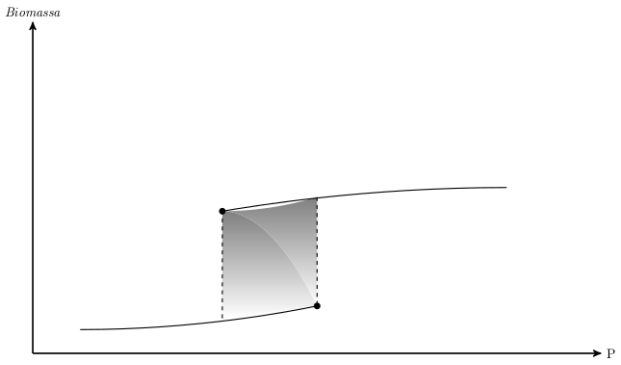


Figura 5.2: Ilustração da ocorrência de estado estável alternativo com histerese. As variáveis de exemplo são as mesmas da Figura 5.1.

Finalmente, a partir das definições de *tipping points*, a avaliação de taxas de recuperação foi proposta para tentar encontrar o momento de aproximação de condições de mudança brusca no regime do sistema e avaliar quanto uma população é resiliente às mudanças no ambiente em que reside. Para isso é realizada uma análise do comportamento

de crescimento de uma população e calculada uma taxa de recuperação, após a realização de perturbações nas quais o tamanho da população é reduzido.

Como ilustração desse processo, apresentamos a Figura 5.3. Neste caso, temos o mesmo cenário de P e Biomassa. O P decresce com variante de estresse e conforme isso acontece, perturbações são aplicadas diminuindo a biomassa. No momento da perturbação, representada pela linha vertical do meio em vermelho, define-se também o valor de referência para onde se desejaria retornar, com a linha vertical da esquerda, em azul. Após isso, mede-se por um determinado intervalo, representado pela linha vertical à direita, roxa, quanto a população cresce (ou decresce), e assim, com a comparação do valor de referência, é obtida a taxa de recuperação para essa perturbação.

Múltiplas perturbações são realizadas, conforme o valor de estresse é aplicado e para cada uma delas é obtida uma taxa de recuperação. Caso as taxas de recuperação sejam descendentes, em teoria, o sistema se aproxima de um *tipping point*, e no caso do ecossistema, poderia estar se aproximando de uma mudança de estado no qual a população entra em extinção.

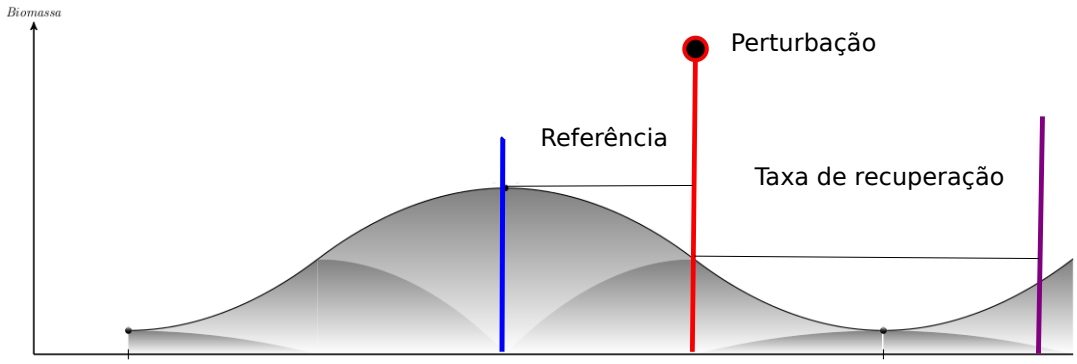


Figura 5.3: Metodologia para calcular a taxa de recuperação de uma população após uma perturbação ser aplicada. A primeira linha vertical na esquerda (azul) indica o tamanho da janela de referência antes de uma perturbação, a linha do meio (vermelha) indica o momento da perturbação, e a linha da direita (roxa) indica o tamanha da janela na qual a taxa de recuperação será medida. As variáveis de exemplo são as mesmas da Figura 5.1.

5.2 Resiliência no EvoCyle

Utilizamos então os conceitos de resiliência ecológica no nosso sistema de vida artificial para ilustrar como estes elementos podem ser aplicados em ecossistemas artificiais. Nesta seção, descrevemos como estes conceitos foram aplicados em maiores detalhes.

Nos capítulos anteriores, apresentamos um simulador de vida artificial que usa inteligência coletiva e algoritmos evolutivos combinados para formar um ecossistema evolutivo com predadores e presas. Como observado nos resultados ao final do Capítulo 4, podemos observar que a dinâmica populacional dos predadores possui considerável relação com a capacidade da população encontrar regiões férteis do espaço de busca. Mais especificamente, notamos que existem situações de extinção da população com regiões de erro alto e estabilidade da população ou crescimento exponencial em regiões de erro baixo. Essas situações podem ser consideradas estados alternativos da população de PSOGP.

5.2.1 Questões Específicas

De frente aos conceitos de resiliência introduzidos, formulamos então as seguintes questões:

QE-5.1 A população predadora PSOGP é resiliente ou não em um dado momento de uma simulação?

QE-5.2 De quais maneiras a informação de resiliência poderia ser utilizada dentro do sistema EvoCycle?

Para responder essas perguntas, tomamos como base as simulações que usam a população de PSOGP predando uma população de ABC, sendo as duas ocupando o espaço de uma função elaborada que fornece energia para a população ABC. A partir deste comportamento inicial propomos utilizar um indicador da categoria *critical slowing down*. Para isso, fazemos a análise das taxas de recuperação após uma série de perturbações na população de predadores. Na próxima seção detalhamos as configurações das simulações.

5.3 Configuração Experimental

Ao realizar as simulações base do EvoCyle, as configurações das populações para os parâmetros do GP, PSO e características de Alife foram as mesmas daquelas indicadas na Tabela 4.1. Para esse modelo de referência, permitimos que a execução continuasse mesmo que um erro baixo da função aplicada fosse encontrado. A execução poderia continuar até que a população superasse 30000 indivíduos ou fosse extinta. A função utilizada para o fornecimento de energia foi a f8 em 5 dimensões, pois nos experimentos anteriores verificamos que o PSOGP foi capaz de encontrar regiões boas do espaço de soluções.

A Tabela 5.1 apresenta as configurações utilizadas para obtermos o indicador de *critical slowing down* de taxa de recuperação. Similar à maneira detalhada na Seção 5.1, perturbações foram realizadas na população de PSOGP. As perturbações foram aplicadas a cada 200 iterações e a avaliação da taxa de recuperação medida em 50 iterações. O valor de referência de retorno da população (*baseline*) é a média do valor da média simples móvel (SMA - *simple moving average*) na qual se encontrava a população nas 30 iterações antes da perturbação. As perturbações consistem em retirar 50% da população de predadores. Além das perturbações, a variante de estresse sobre os predadores considerada é a quantidade de colmeias, das quais 2 colmeias aleatórias são retiradas, também a cada 200 iterações.

A SMA é uma média aritmética móvel dos indivíduos da população, na qual é calculada uma média em um determinado intervalo de tempo para cada novo ponto da SMA. Para calcular a SMA da população, a seguinte expressão foi utilizada:

$$SMA = \frac{\sum_{i-W}^{i} P(i)}{W} \tag{5.1}$$

em que i é uma determinada iteração, W é o tamanho do janela de intervalo que será feita a média e $P(i)$ é o número de indivíduos da população na iteração i. Dessa forma,

Tabela 5.1: Configuração do estudo com perturbações na população de PSOGP.

População máxima	30000
Perturbações nas iterações	×200
Perturbação Efeito	-50%
Estresse na presa	-2
Janela de observação (W_{rec})	50
referência base	SMA
SMA W	3
Janela de referência (W_{base})	30

o *baseline* assume a média de SMAs nas W_{base} iterações anteriores:

$$base = \frac{\sum_{i-Wbase}^{i} SMA(i)}{Wbase} \tag{5.2}$$

Assim, a cada vez que uma perturbação era aplicada na população, a SMA era calculada naquela iteração com uma janela de tamanho $W = 3$. Após isso, foi observado o comportamento da população em uma janela de $W_{rec} = 50$ iterações. Nesse período, verificamos se a população voltava a crescer em direção ao valor da média de *baseline* obtido na iteração da perturbação, e mais especificamente em que velocidade isso acontecia. Essa velocidade chamamos de taxa de recuperação e o definimos como RR, da seguinte forma:

$$\frac{dP(i)}{di} = -RR(base - P(i)) \tag{5.3}$$

em que $P(i)$ é o número de indivíduos em uma dada iteração i. A taxa de recuperação RR após uma perturbação é obtida através de uma regressão linear na curva obtida pela diferença do *baseline* e o valor da população com i variando entre o momento de uma perturbação e o tamanha de uma janela desejada.

5.4 Resultados e Discussão

Na Figura 5.4, apresentamos o crescimento populacional que foi utilizado como base, em uma simulação da população de PSOGP predando colmeias ABC. Nela podemos ver as iterações representadas no eixo x e o número de indivíduos da população no eixo y. Neste caso, a simulação executou mais de duas mil iterações com uma taxa de natalidade similar a de mortalidade, até que a população atingiu um ponto de instabilidade e passa a crescer exponencialmente.

Após isso, utilizamos o resultado dessa simulação base e realizamos a aplicação de perturbações na população de predadores, seguindo os parâmetros estabelecidos na Tabela 5.1. Os resultados das simulações com perturbações são apresentados nas Figuras 5.5 e 5.6. As iterações e número de indivíduos são similares à simulação base. As linhas verticais que cruzam o gráfico representam a janela utilizada para analisar as taxas de recuperação. Em cada uma das perturbações, há uma linha (vermelha) e a linha que se

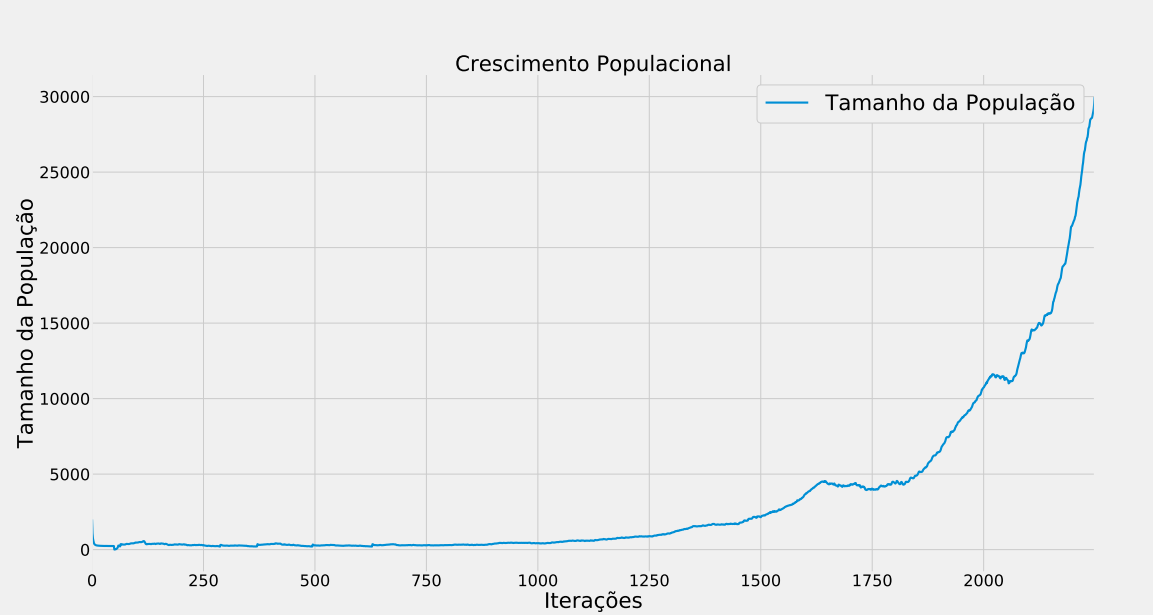


Figura 5.4: Crescimento populacional no processo de predação PSOGP × ABC.

Tabela 5.2: Taxas de recuperação após sucessivas perturbações.

Cenário	200	400	600	800	1000	1200	1400	1600
Exponencial	1.773	1.502	0.327	1.646	1.186	22.576	47.152	-
Colapso	0.446	5.236	0.899	-5.470	0.805	2.421	-1.559	-0.982

encontra próxima a ela a sua direita (roxa), é a iteração do tamanho da janela. Para cada um dos intervalos também é apresentada uma curva da regressão linear da taxa de recuperação.

Dentre alguns resultados obtidos nessas simulações, dois cenários se destacam para investigarmos a mudança de estados na população de PSOGP. Nos resultados apresentados na Figura 5.5, mesmo com o uso das perturbações, a população se mostrou resiliente, apresentando crescimento populacional similar ao cenário sem perturbações. Chamaremos este cenário de "Exponencial". Por outro lado, no cenário apresentado na Figura 5.6, após perturbações consecutivas, a população de predadores colapsou para extinção. Este cenário será denominado "Colapso".

As taxas de recuperação nos dois cenários são apresentadas na Tabela 5.2.

Utilizando estes valores de taxa de recuperação, quando analisamos o comportamento da curva em relação à variável de estresse, que é a quantidade de colmeias disponíveis na simulação, podemos identificar dois resultados muito distintos de acordo com as situações em que houve ou não colapso da população de predadores. As Figuras 5.7 e 5.8 retratam essa configuração. Nelas podemos ver que as taxas de recuperação apresentam uma tendência de subida nos casos das simulações do cenário "Exponencial", e podemos ver que as taxas de recuperação apresentam uma tendência de queda no cenário "Colapso".

Com estes resultados, respondemos a questão QE-5.1. As taxas de recuperação (um indicador da resiliência da população da classe *critical slowing down*) sugerem que ao

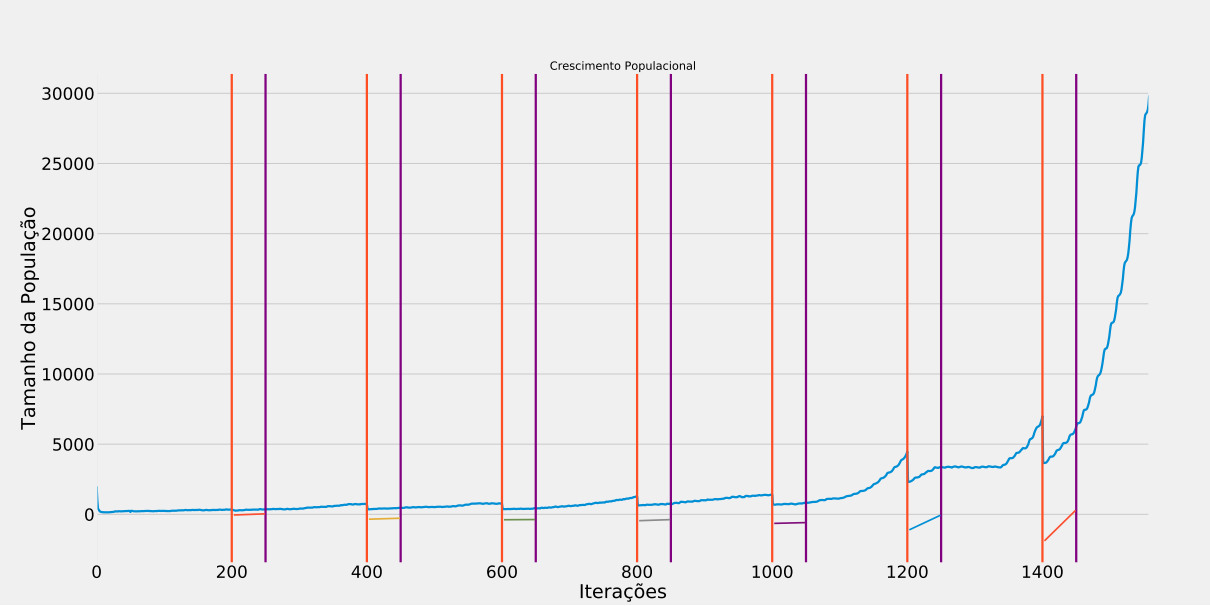


Figura 5.5: Crescimento Populacional no processo de predação com perturbações a cada 200 iterações. Neste caso a população apresenta um crescimento exponencial e a execução é interrompida

apresentar uma tendência crescente, a população de PSOGP se encontra estável no espaço de busca, uma vez que foi resiliente às perturbações aplicadas mesmo diante da queda de colmeias disponíveis. Da mesma forma, no cenário de Colapso, a população não se mostrou resiliente às perturbações e isso pode também ser identificado pela tendência de queda nas taxas de recuperação.

Em relação à questão QE-5.2, uma forma que a informação de resiliência da população poderia ser utilizada no EvoCycle seria a verificação das taxas de recuperação no processo de simulação. Periodicamente, as taxas de recuperação da população PSOGP poderiam ser mensuradas para se avaliar a tendências de crescimento ou não.

No caso da aplicação de otimização contínua, o crescimento exponencial da população pode ocorrer antes que se encontre uma região de erro baixo, como quando ocorre uma prisão em mínimos locais. Quanto aos casos de colapso, a tendência de queda poderia ser utilizada para prevenir a extinção da população e assim continuar com o processo de busca. Para isso, um fator de dispersão poderia ser empregado, fazendo com que a população passe a visitar outras localidades.

Finalmente, a exploração da informação de resiliência da população poderia ser empregada no estudo de vida artificial para validar conceitos e obter informações sobre o sistema. Por exemplo, sabendo que a população de PSOGP é resiliente em um determinado momento, com a inserção de uma espécie predadora, de PSOGP ou ABC, poderíamos realizar mais um estudo ecológico das consequências dessa interação. Este inclusive poderia ser o fator de dispersão da população de PSOGP para melhorar os resultados de busca. Outro exemplo seria a inserção de uma outra espécie de presas para o PSOGP, que funcionaria como um modelo ecológico de substituição, no qual seria simulado um cenário em que uma população PSOGP se encontra em direção a extinção, em decorrência da decadência de uma das populações de presas (fator de estresse), mas uma nova espécie introduzida

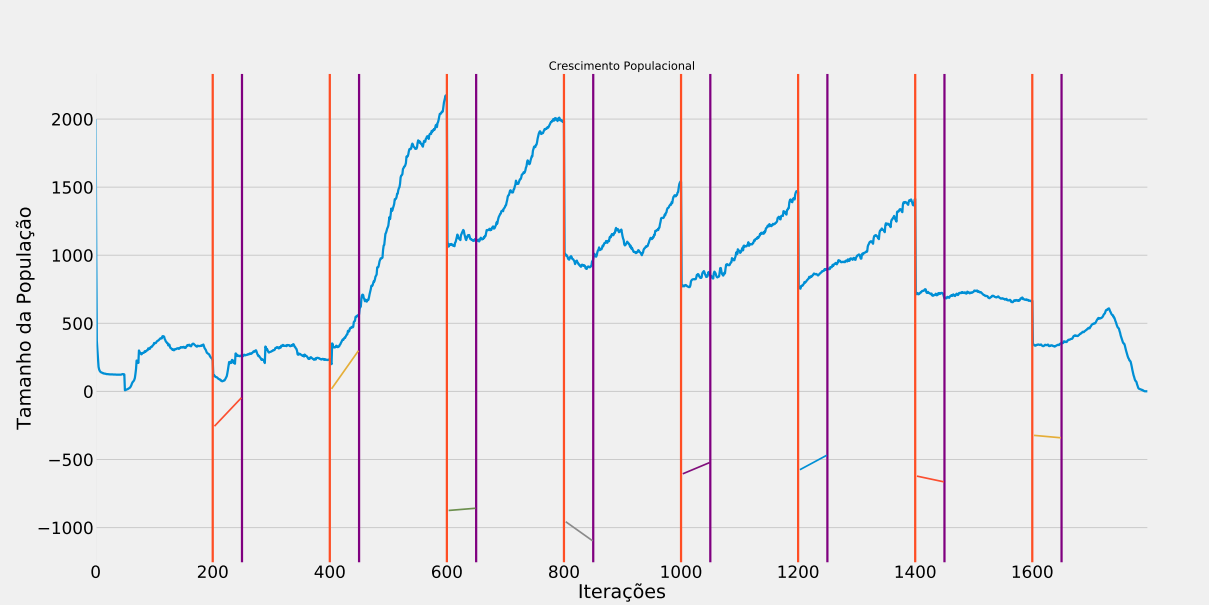


Figura 5.6: Crescimento Populacional no processo de predação com perturbações a cada 200 iterações. Neste caso ocorreu o colapso da população.

restauraria a resiliência da população de predadores.

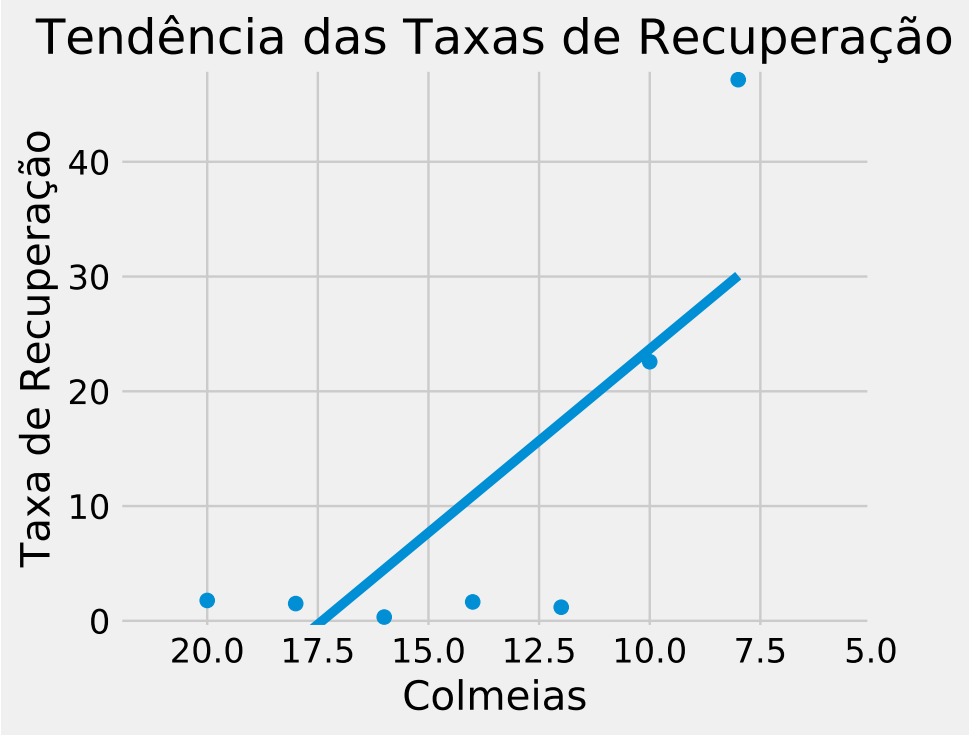


Figura 5.7: Tendência das taxas de recuperação no cenário em que ocorre uma mudança de estado para o crescimento populacional exponencial.

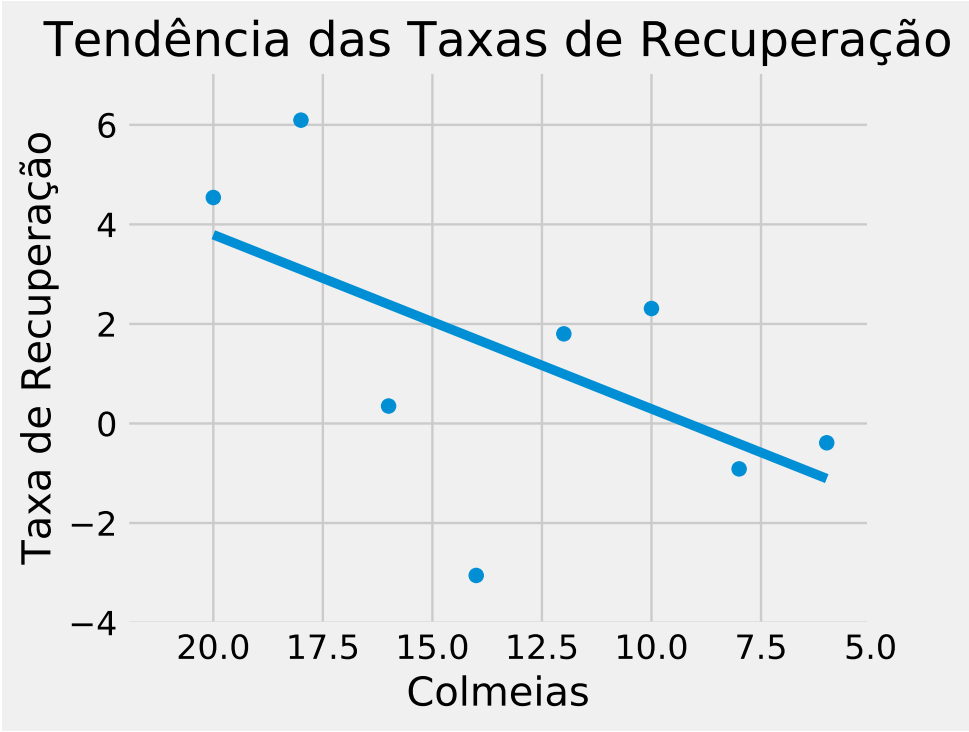


Figura 5.8: Tendência das taxas de recuperação no cenário em que ocorre o colapso da população.

Capítulo 6

Conclusões

Neste trabalho, apresentamos o *EvoCycle*, um arcabouço para simulação de Vida Artificial em um ecossistema de meta-heurísticas de Computação Evolutiva. Com o objetivo de incorporar aspectos da teoria da evolução que não são costumeiramente utilizados em computação evolutiva, como a interação de indivíduos a partir de relacionamentos ecológicos e a avaliação da resiliência de populações, fomos capazes de realizar simulações de vida artificial e identificar o surgimento de padrões de busca no nosso modelo. Em seguida, apresentamos as principais contribuições obtidas e apontamos caminhos para serem trilhados em trabalhos futuros.

6.1 Contribuições

A primeira principal contribuição reside na concepção de um modelo computacional que não realiza a seleção de indivíduos baseada em uma avaliação direta de função de *fitness*. O arcabouço EvoCycle é organizado em três camadas: genética, comportamento coletivo, e ecológica. Nele, diferentes populações de indivíduos (na camada coletiva) evoluem ao longo do tempo baseados em como sua codificação genética os governa (camada genética), e sujeitos a pressões seletivas exercidas por relacionamentos ecológicos (camada ecológica). Com tal arquitetura, aumentamos a plausibilidade biológica do processo evolutivo em nossas simulações de vida artificial e na realização de computação natural como um todo.

Após um levantamento bibliográfico que respondia quais trabalhos se relacionam a esta pesquisa (questão Q-1), foi implementada uma prova de conceito para avaliar se uma população teria sucesso em evoluir dentro do nosso modelo (questão Q-2).

Para responder a questão Q-2, foi necessária uma avaliação em duas etapas. Na primeira etapa, formulada como a questão QE-3.1, utilizamos uma população de híbridos de PSO na segunda camada, codificados por árvores GP na camada genética, sendo os relacionamentos ecológicos definidos como: alimentação do meio ambiente, envelhecimento e reprodução. Dessa forma, a pressão seletiva ocorre de forma sexual. Esta prova de conceito foi aplicada nas funções Esfera, Rosenbrock, Sharp Ridge, Rastrigin e Gallagher 101 peaks para fins de avaliação.

Dos resultados obtidos, podemos concluir que no nosso modelo, ao longo do processo evolutivo, o fenótipo de busca local obteve mais sucesso em estágios iniciais. Em está-

gios mais avançados entretanto, uma estratégia de movimentar-ficar-comer foi mais bem sucedida. Também pudemos observar que as frequências de genótipos variaram ao longo da execução a partir dos relacionamentos ecológicos estabelecidos. De fato, em diferentes estágios da execução, nichos fenotípicos específicos foram gerados pelo arcabouço. Em resumo, respondendo a primeira etapa da questão Q-2, verificamos que a seleção sexual oriunda da movimentação das partículas teve reflexo nos genes da população (questão QE-3.1).

Em relação ao processo de otimização (questão Q-3), também fizemos uma verificação em duas partes. Na primeira, formulada como a questão QE-3.2, as soluções evoluídas tiveram mais sucesso em encontrar regiões próximas do valor objetivo em baixas dimensões. As estratégias de busca obtidas foram comparáveis a uma busca utilizando o PSO clássico, o que responde a questão QE-3.2 de forma afirmativa. No entanto, em altas dimensões esses resultados não se mostraram escaláveis. Isto pode ser explicado pelo grande número de parâmetros do EvoCycle, a uma tendência em ficar preso a mínimos locais, e também a dificuldade em estabelecer um critério de proximidades em alta dimensionalidade.

A segunda grande contribuição foi a realização de simulações de vida artificial em um ecossistema construído com diferentes espécies de algoritmos baseados em inteligência coletiva, e que se relacionam por meio da predação. Nestas simulações, a hibridização de PSOGP se alimenta de colmeias ABC. As mesmas funções foram utilizadas para estabelecer o ambiente de vida artificial. Com esta contribuição, realizamos a segunda etapa das avaliações para as questões Q-2 e Q-3 e assim foi possível responde-las por completo.

Nas simulações, descobrimos que as interações das populações de PSOGP e ABC através da predação levaram ao surgimento de padrões de movimento na população de PSOGP. Com a análise da variação de genótipos específicos ao longo das iterações da simulação, descobrimos que a movimentação em direção ao melhor global foi dominante em diferentes cenários (questões QE-4.1 e QE-4.2). Também foi possível verificar que a troca de informações entre as duas populações foi crucial para a sobrevivência da população de predadores (questão QE-4.3). E assim, respondemos por completo a questão Q-2, ao certificar a influência de mais de uma interação ecológica nos genes da população.

Além disso, essa troca de informação também favoreceu a população de ABC a encontrar melhores resultados em sua busca. Sendo assim, respondemos a questão Q-3 afirmativamente, pois a partir do conjunto dos resultados em otimização, vimos que o EvoCycle tem sucesso em encontrar soluções válidas, mostrando-se promissor para exploração futura.

Finalmente, a última contribuição reside na introdução dos conceitos de resiliência de populações aplicados em simulações de vida artificial e em aplicações de computação evolutiva, para responder a questão Q-4. A partir da análise das taxas de recuperação após perturbações, fomos capazes de determinar se uma população de predadores PSOGP era ou não resiliente (questão QE-5.1). Com essa informação, poderíamos construir populações estáveis em modelos de vida artificial e até mesmo melhorar o processo de otimização ao impedir que as populações sejam extintas (QE-5.2).

Ao longo do período de doutoramento, os resultados citados foram apresentados no Simpósio da sociedade japonesa de computação evolutiva em 2017[1], no formato de pôs-

[1] `http://www.jpnsec.org/symposium201703.html` (Visitado em Setembro 2021).

ter, com o título *Ecology-aware evolutionary computation systems*; Uma submissão foi realizada para periódico internacional.

6.2 Trabalhos Futuros

Os seguintes tópicos são elencados como possibilidades de trabalhos futuros:

1. Uma possibilidade de trabalho futuro diz respeito à extensão do ecossistema. Isso poderia ser realizado a partir da inclusão de outros algoritmos de busca para servir como modelos de espécie no sistema. Além disso, outros algoritmos evolutivos também poderiam ser usados para codificação genética. Como exemplo, poderíamos utilizar GE para a codificação do mesmo PSO e obter mais opções de variação genética. Outro exemplo seria o uso de GAs codificando ACO, uma vez que as tabelas construídas no ACO poderiam ser uma codificação das cadeias de bits de GAs. Com a presença de mais espécies, outros relacionamentos ecológicos poderiam ser incluídos, como competição, simbiose ou parasitismo. Diante de um ecossistema mais complexo, os estudos de vida artificial poderiam ser aprofundados e até mesmo validações de estudos ecológicos poderiam ser realizados.

2. Utilizamos nesse trabalho o cenário de otimização contínua para situar o ambiente virtual do EvoCycle. No entanto, outras aplicações poderiam se beneficiar do *framework* proposto. A primeira sugestão nesse sentido seria o cenário de otimização multi objetivo. Uma vez que temos múltiplas populações buscando em um mesmo espaço, as relações ecológicas poderiam ser estabelecidas de forma que cada uma delas se dirija a objetivos diferentes.

3. O EvoCycle é um arcabouço para a utilização de diferentes meta heurísticas ao mesmo tempo, cada uma delas podendo ser composta por algoritmos de SI e EA. Como consequência, existem muitos parâmetros em cada um destes algoritmos. Assim, outro trabalho futuro consiste na exploração paramétrica, investigando valores e estratégias que poderiam contribuir em aspectos de desempenho.

4. Um caminho promissor consiste na validação do *framework* proposto em outras aplicações. Em particular, vislumbra-se a utilização do *framework* proposto em problemas de busca de imagens, como busca de imagem por conteúdo (do inglês, *Content-based Image Retrieval* – CBIR). Sabidamente, a EC foi utilizada para a combinação e seleção de características no cenário de buscas de imagens [5]. Como fomos capazes de identificar genes pertinentes para o processo de otimização contínua, o processo evolutivo proposto poderia ajudar na identificação de características relevantes em CBIR.

5. Nesta mesma linha, sugere-se ainda a investigar o uso do arcabouço proposto em tarefas de busca multimodal [14]. Neste caso, diferentes populações representam tipos de informação diferente. Os diferentes tipos de informação na busca multimodal usualmente são representados por características de baixo nível, de forma

semelhante a CBIR. Com o uso do EvoCycle, no final do processo evolutivo teríamos informação de quais dessas características seriam mais relevantes.

6. Com o *framework* estabelecido, outro possível trabalho futuro é um estudo ecológico a partir da modelagem da vida artificial presente no EvoCycle. Dessa maneira, a partir de uma certa configuração inicial de populações e relações presentes, a validação de conceitos de Ecologia poderia ser explorada. Por exemplo, com duas populações com número de indivíduos flutuantes e a relação de predação entre elas, poderíamos validar se o número de predadores acompanha o de presas.

7. Por fim, propomos que o conceito de resiliência introduzido no estudo de EC e AL seja propriamente implementado, como abordado no Capítulo 5. Propomos que as taxas de recuperação de populações de MHs sejam utilizadas para auxiliar no processo de busca em EC. Similarmente, outros indicadores de resiliência poderiam ser aplicados ou usados em conjunto para refinar a busca de soluções. Além disso, os conceitos de resiliência também poderiam ser validados em estudos ecológicos, similares ao estudo mencionado no item anterior.

Referências Bibliográficas

[1] Moein Fazeli Hassan Abadi and Hassan Rezaei. Data clustering using hybridization strategies of continuous ant colony optimization, particle swarm optimization and genetic algorithm. *British Journal of Mathematics & Computer Science*, 6(4):336, 2015.

[2] Chris Adami and C Titus Brown. Evolutionary learning in the 2d artificial life system 'avida'. In *Artificial life IV*, volume 1194, pages 377–381. MIT press Cambridge, MA, 1994.

[3] Christoph Adami, Jason Moore, Fred Dyer, Arend Hintze, and Randal Olson. Exploring the coevolution of predator and prey morphology and behavior. In *Artificial Life Conference Proceedings 13*, pages 250–257. MIT Press, 2016.

[4] Manuel Alfonseca and Francisco José Soler Gil. Evolving a predator–prey ecosystem of mathematical expressions with grammatical evolution. *Complexity*, 20(3):66–83, 2015.

[5] Felipe SP Andrade, Jurandy Almeida, Hélio Pedrini, and Ricardo da S Torres. Fusion of local and global descriptors for content-based image and video retrieval. In *Iberoamerican Congress on Pattern Recognition*, pages 845–853. Springer, 2012.

[6] Andrew M Assad and Norman H Packard. Emergent colonization in an artificial ecology. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life,(ed.) FJ Varela and P. Bourgine*, pages 143–152, 1992.

[7] Gayle C Avery and Harald Bergsteiner. Sustainable leadership practices for enhancing business resilience and performance. *Strategy & Leadership*, 2011.

[8] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. Handbook of evolutionary computation. *Release*, 97(1):B1, 1997.

[9] Theodore C Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 114–121. Morgan Kaufmann, 1995.

[10] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[11] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287, 2009.

[12] Elhanan Borenstein, Isaac Meilijson, and Eytan Ruppin. The effect of phenotypic plasticity on evolution in multipeaked fitness landscapes. *Journal of evolutionary biology*, 19(5):1555–1570, 2006.

[13] Jason Brownlee. Complex adaptive systems. Technical report, Swinburne University of Technology, Melbourne, Australia, 2007.

[14] Rodrigo Tripodi Calumby, Ricardo da Silva Torres, and Marcos André Gonçalves. Multimodal retrieval with relevance feedback based on genetic programming. *Multimedia tools and applications*, 69(3):991–1019, 2014.

[15] Xianshun Chen, Yew-Soon Ong, Meng-Hiot Lim, and Kay Chen Tan. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation*, 15(5):591–607, 2011.

[16] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[17] Dara Curran and Colm O'Riordan. Increasing population diversity through cultural learning. *Adaptive Behavior*, 14(4):315–338, 2006.

[18] Charles Darwin. *On the Origin of Species by Means of Natural Selection, Or, The Preservation of Favoured Races in the Struggle for Life*. J. Murray, 1859.

[19] Kedar Nath Das and Raghav Prasad Parouha. Engineering design optimization using hybrid (de-pso-de) algorithm. In *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*, pages 461–475. Springer, 2015.

[20] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.

[21] Richard Dawkins. *The Selfish Gene*. Oxford paperbacks. Oxford University Press, 1989.

[22] Leandro Nunes de Castro. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1–36, 2007.

[23] Andrea De Lorenzo, Alberto Bartoli, Mauro Castelli, Eric Medvet, and Bing Xue. Genetic programming in the twenty-first century: a bibliometric and content-based analysis from both sides of the fence. *Genetic Programming and Evolvable Machines*, 21(1):181–204, 2020.

[24] Javier Del Ser, Eneko Osaba, Daniel Molina, Xin-She Yang, Sancho Salcedo-Sanz, David Camacho, Swagatam Das, Ponnuthurai N Suganthan, Carlos A Coello Coello, and Francisco Herrera. Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, 48:220–250, 2019.

[25] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, Feb 1996.

[26] Russ C Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 1, pages 39–43. New York, NY, 1995.

[27] Agoston E. Eiben and Jim Smith. From evolutionary computation to the evolution of things. *Nature*, 521(7553):476–482, May 2015. Insight.

[28] Mohammed El-Abd and Mohamed S. Kamel. Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2269–2274, New York, NY, USA, 2009. ACM.

[29] Ted H Emigh. A comparison of tests for hardy-weinberg equilibrium. *Biometrics*, pages 627–642, 1980.

[30] Michael G Epitropakis, Vassilis P Plagianakos, and Michael N Vrahatis. Evolving cognitive and social experience in particle swarm optimization through differential evolution: a hybrid approach. *Information Sciences*, 216:50–92, 2012.

[31] Lawrence J Fogel, Alvin J Owens, and Michael J Walsh. *Artificial intelligence through simulated evolution*. Wiley, Chichester, WS, UK, 1966.

[32] Stephanie Forrest and Terry Jones. Modeling complex adaptive systems with echo. In *Complex Systems: Mechanisms of Adaptation*, pages 3–21. IOS Press, 1994.

[33] David Forsyth and Jean Ponce. *Computer vision: A modern approach*. Prentice hall, 2011.

[34] John H Gillespie. *Population Genetics: A Concise Guide*. Population Genetics. Johns Hopkins University Press, 2010.

[35] Robin Gras, Didier Devaurs, Adrianna Wozniak, and Adam Aspinall. An individual-based evolving predator-prey ecosystem simulation using a fuzzy cognitive map as the behavior model. *Artificial life*, 15(4):423–463, 2009.

[36] Yun-li Gu, Xin Xu, Jie Du, and Huan-yan Qian. Optimization algorithm based on artificial life algorithm and particle swarm optimization. In *2009 Second International Conference on Information and Computing Science*, volume 3, pages 173–176. IEEE, 2009.

[37] Tong Guo, Changgeng Li, Jinglei Yang, Pengfei Wang, Jianling Yue, Xiaozhong Huang, Jing Wang, and Xiu-Zhi Tang. Holey, anti-impact and resilient thermoplastic urethane/carbon nanotubes fabricated by a low-cost "vapor induced phase separation" strategy for the detection of human motions. *Composites Part A: Applied Science and Manufacturing*, 136:105974, 2020.

[38] John Burdon Sanderson Haldane. A mathematical theory of natural and artificial selection, part V: Selection and mutation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 23(07):838–844, 7 1927.

[39] Natsuki Higashi and Hitoshi Iba. Particle swarm optimization with gaussian mutation. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 72–79. IEEE, 2003.

[40] W Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, 1990.

[41] Geoffrey E Hinton and Steven J Nowlan. How learning can guide evolution. *Complex systems*, 1(3):495–502, 1987.

[42] Marina Hirota, Milena Holmgren, Egbert H Van Nes, and Marten Scheffer. Global resilience of tropical forest and savanna to critical transitions. *Science*, 334(6053):232–235, 2011.

[43] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[44] John H. Holland. Complex adaptive systems. *Daedalus*, 121(1):17–30, 1992.

[45] Essam H Houssein, Ahmed G Gad, Kashif Hussain, and Ponnuthurai Nagaratnam Suganthan. Major advances in particle swarm optimization: Theory, analysis, and application. *Swarm and Evolutionary Computation*, 63:100868, 2021.

[46] Julian Huxley, Massimo Pigliucci, and GB Müller. *Evolution: The Modern Synthesis: The Definitive Edition*. Mit Press, 2010.

[47] INRIA. Comparing continuous optimisers: Coco. http://coco.gforge.inria.fr/doku.php, 2020.

[48] Takashi Ito, Marcin L Pilat, Reiji Suzuki, and Takaya Arita. Population and evolutionary dynamics based on predator–prey relationships in a 3d physical simulation. *Artificial life*, 22(2):226–240, 2016.

[49] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471, Nov 2007.

[50] James Kennedy and Rui Mendes. Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1671–1676 vol.2, 2002.

[51] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[52] C.G. Langton. *Artificial Life: An Overview*. A Bradford book. BRADFORD BOOK, 1997.

[53] Christopher G. Langton. *Artificial Life: Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[54] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[55] Timothy Lenton, Valerie Livina, Vasilis Dakos, Egbert H. Van Nes, and Marten Scheffer. Early warning of climate tipping points from critical slowing down: comparing methods to improve robustness. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 370(1962):1185–1204, 2012.

[56] Yuhua Li, Zhi-Hui Zhan, Shujin Lin, Jun Zhang, and Xiaonan Luo. Competitive and cooperative particle swarm optimization with information sharing mechanism for global optimization problems. *Information Sciences*, 293:370–382, 2015.

[57] Morten Lovbjerg, Thomas Kiel Rasmussen, and Thiemo Krink. Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the genetic and evolutionary computation conference*, volume 2001, pages 469–476. Citeseer, 2001.

[58] Manuel Lozano and Carlos García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Computers & Operations Research*, 37(3):481–497, 2010.

[59] Zhanshan(Sam) Ma. Towards a population dynamics theory for evolutionary computing: Learning from biological population dynamics in nature. In *Artificial Intelligence and Computational Intelligence*, volume 5855 of *Lecture Notes in Computer Science*, pages 195–205. Springer Berlin Heidelberg, 2009.

[60] Gregor Mendel. Experiments in Plant Hybridization. *Proceedings of the Brünn Natural History Society*, IV, 1865.

[61] Seyedali Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, pages 1–21, 2016.

[62] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 2014.

[63] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[64] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[65] Robin R Murphy. *Introduction to AI robotics*. MIT press, 2019.

[66] Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[67] Norman Packard, Mark A Bedau, Alastair Channon, Takashi Ikegami, Steen Rasmussen, Kenneth O Stanley, and Tim Taylor. An overview of open-ended evolution: Editorial introduction to the open-ended evolution ii special issue. *Artificial life*, 25(2):93–103, 2019.

[68] Ingo Paenke, Yaochu Jin, and Jürgen Branke. Balancing population-and individual-level adaptation in changing environments. *Adaptive Behavior*, 17(2):153–174, 2009.

[69] Rafael S Parpinelli and Heitor S Lopes. New inspirations in swarm intelligence: a survey. *International Journal of Bio-Inspired Computation*, 3(1):1–16, 2011.

[70] Rafael S Parpinelli and Heitor S Lopes. Biological plausibility in optimisation: an ecosystemic view. *International Journal of Bio-Inspired Computation*, 4(6):345–358, 2012.

[71] Rafael Stubs Parpinelli and Heitor Silvério Lopes. A computational ecosystem for optimization: review and perspectives for future research. *Memetic Computing*, 7(1):29–41, 2015.

[72] Rodrigo Pasti, Fernando José Von Zuben, and Leandro Nunes de Castro. Ecosystems computing: Introduction to biogeographic computation. *International Journal of Natural Computing Research (IJNCR)*, 2(4):47–67, 2011.

[73] Riccardo Poli, William B Langdon, and Owen Holland. Extending particle swarm optimisation via genetic programming. In *European Conference on Genetic Programming*, pages 291–300. Springer, 2005.

[74] Jacek Rak, David Hutchison, Janos Tapolcai, Rasa Bruzgiene, Massimo Tornatore, Carmen Mas-Machuca, Marija Furdek, and Paul Smith. Fundamentals of communication networks resilience to disasters and massive disruptions. In *Guide to Disaster-Resilient Communication Networks*, pages 1–43. Springer, 2020.

[75] Holt Rinehart and Winston. *Modern Biology: Investigating Microbes (with Living Materials) Science Kit*. Mod Biology 2009. Houghton Mifflin School, 2009.

[76] Luis Mateus Rocha. Contextual genetic algorithms: Evolving developmental rules. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 368–382, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[77] S. Russell, S.J. Russell, P. Norvig, and E. Davis. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010.

[78] Shahrzad Saremi, Seyedeh Zahra Mirjalili, and Seyed Mohammad Mirjalili. Evolutionary population dynamics and grey wolf optimizer. *Neural Computing and Applications*, pages 1–7, 2014.

[79] Taiji Satoh, Yoshiki Mizukami, Kanya Tanaka, and Koichi Nara. A two-level alife system with predator. *Electronics and Communications in Japan (Part II: Electronics)*, 87(8):53–60, 2004.

[80] Marten Scheffer, J Elizabeth Bolhuis, Denny Borsboom, Timothy G Buchman, Sanne MW Gijzel, Dave Goulson, Jan E Kammenga, Bas Kemp, Ingrid A van de Leemput, Simon Levin, et al. Quantifying resilience of humans and other animals. *Proceedings of the National Academy of Sciences*, 115(47):11883–11890, 2018.

[81] Marten Scheffer, Stephen R Carpenter, Timothy M Lenton, Jordi Bascompte, William Brock, Vasilis Dakos, Johan Van de Koppel, Ingrid A Van de Leemput, Simon A Levin, Egbert H Van Nes, et al. Anticipating critical transitions. *science*, 338(6105):344–348, 2012.

[82] Marten Scheffer, Steve Carpenter, Jonathan A Foley, Carl Folke, and Brian Walker. Catastrophic shifts in ecosystems. *Nature*, 413(6856):591–596, 2001.

[83] Dan Simon. Biogeography-based optimization. *Evolutionary Computation, IEEE Transactions on*, 12(6):702–713, 2008.

[84] Ronald A Fisher Sir and Henry Bennett. *The Genetical Theory of Natural Selection: A Complete Variorum Edition*. OUP Oxford, 1930.

[85] Andrew N Sloss and Steven Gustafson. 2019 evolutionary algorithms review. *Genetic Programming Theory and Practice XVII*, page 307, 2020.

[86] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[87] Peter Sunehag, Guy Lever, Siqi Liu, Josh Merel, Nicolas Heess, Joel Z Leibo, Edward Hughes, Tom Eccles, and Thore Graepel. Reinforcement learning agents acquire flocking and symbiotic behaviour in simulated ecosystems. In *The 2018 Conference on Artificial Life*, pages 103–110. MIT Press, 2019.

[88] Muhammad Rizwan Tanweer, Sundaram Suresh, and Narasimhan Sundararajan. Self regulating particle swarm optimization algorithm. *Information Sciences*, 294:182–202, 2015.

[89] Robert J Taylor. *Predation*. Population and Community Biology. Springer Netherlands, 2013.

[90] Kurt Thearling and Thomas S Ray. Evolving multi-cellular artificial life. In *Artificial Life IV*, pages 283–288, 1994.

[91] Kirsten Thonicke, Fanny Langerwisch, Matthias Baumann, Pedro J Leitão, Tomáš Václavík, Ane Alencar, Margareth Simões, Simon Scheiter, Liam Langan, Mercedes Bustamante, et al. A social-ecological approach to identify and quantify biodiversity tipping points in south america's seasonal dry ecosystems. *Biogeosciences Discussions*, pages 1–22, 2019.

[92] Alan M. Turing. Intelligent machinery. In Bernard Meltzer and Donald Michie, editors, *Machine Intelligence*, volume 5, chapter 1, pages 3–23. Edinburgh University Press, Edinburgh, UK, 1969.

[93] Egbert H van Nes, Marina Hirota, Milena Holmgren, and Marten Scheffer. Tipping points in tropical tree cover: linking theory to data. *Global change biology*, 20(3):1016–1021, 2014.

[94] Srinivasa Shivakar Vulli and Sanjeev Agarwal. Individual-based artificial ecosystems for design and optimization. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, GECCO '08, pages 273–280, New York, NY, USA, 2008. ACM.

[95] Pu Wang, Ke Tang, Edward PK Tsang, and Xin Yao. A memetic genetic programming with decision tree-based local search for classification problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 917–924. IEEE, 2011.

[96] Darrell Whitley, Soraya Rana, and Robert B Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of computing and information technology*, 7(1):33–47, 1999.

[97] Olaf Witkowski and Takashi Ikegami. How to make swarms open-ended? evolving collective intelligence through a constricted exploration of adjacent possibles. *Artificial life*, 25(2):178–197, 2019.

[98] Sewall Wright. *Statistical Genetics in Relation to Evolution*. Actualités scientifiques et industrielles. Hermann, 1939.

[99] Bin Xin, Jie Chen, Juan Zhang, Hao Fang, and Zhi-Hong Peng. Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(5):744–767, 2012.

[100] Bo-Suk Yang. Artificial life optimization algorithm and applications. In *Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies*, pages 423–449. IGI Global, 2009.

[101] Chen Yang, Hao Ye, Jing-Chun Wang, and Ling Wang. An artificial life and genetic algorithm based on optimization approach with new selecting methods. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 2, pages 684–688. IEEE, 2002.

[102] Danial Yazdani, Babak Nasiri, Alireza Sepas-Moghaddam, and Mohammad Reza Meybodi. A novel multi-swarm algorithm for optimization in dynamic environments based on particle swarm optimization. *Applied Soft Computing*, 13(4):2144–2158, 2013.

[103] Hai-fei YU and Ding-wei WANG. Food-chain algorithm and comparison of its performance with real-coded ga. *Systems Engineering-Theory & Practice*, 27(6):143–148, 2007.

[104] Yudong Zhang, Shuihua Wang, and Genlin Ji. A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*, 2015:1–38, 2015.