Tutorial Sobre o Uso de Técnicas para Controle Online de Parâmetros em Algoritmos de Inteligência de Enxame e Computação Evolutiva

Leanderson André [1]
Rafael Stubs Parpinelli [1]

Resumo:

A natureza tem sido uma grande fonte de inspiração para o desenvolvimento de abordagens computacionais para otimização. Dois grandes grupos que representam esta classe de algoritmos biologicamente inspirados são a Inteligência de Enxame e a Computação Evolutiva. Tais algoritmos são chamados de metaheurísticas e são reconhecidos como abordagens eficientes para resolução de problemas complexos. Tanto os algoritmos da Inteligência de Enxame como os da Computação Evolutiva compartilham características comuns como a utilização de componentes estocásticos durante o processo de otimização e variados parâmetros de configuração. O ajuste dos parâmetros de um algoritmo possui um papel importante por definirem seu comportamento, guiando a busca e, consequentemente, interferindo na qualidade das soluções encontradas. Porém, o ajuste dos parâmetros não é uma tarefa simples, se tornando um problema de otimização dentro do problema sendo otimizado. Além disso, uma configuração adequada para os parâmetros pode se alterar durante o processo de otimização. Existem duas maneiras de se ajustar os parâmetros de um algoritmo. O ajuste *offline* que é realizado antes da execução do algoritmo e os valores dos parâmetros se mantém fixos, e o controle *online* onde os valores dos parâmetros podem mudar durante o processo de otimização. Este artigo tem foco em revisar as estratégias de controle *online* de parâmetros aplicados nos principais algoritmos da Computação Evolutiva e da Inteligência de Enxame. Como resultado, esta revisão analisa e pontua as principais técnicas e algoritmos utilizados e sugere algumas direções para pesquisas futuras.

Palavras chave: Ajuste de Parâmetros, Computação Evolutiva, Inteligência de Enxame, Controle *online*

[1]Programa de Pós-Graduação em Computação Aplicada,
Universidade do Estado de Santa Catarina, Joinville - SC, Brasil
{leanderson.andre@gmail.com,rafael.parpinelli@udesc.br}

Abstract: Nature has always been a great source of inspiration for the development of computational approaches for optimization. Two major groups representing this class of biologically inspired algorithms are Swarm Intelligence and Evolutionary Computation. Such algorithms are called metaheuristics and are recognized to be efficient approaches for solving complex problems. Both Swarm Intelligence and Evolutionary Computation share common features such as the use of stochastic components during the optimization process and various parameters for configuration. The setup of parameters of an algorithm has an important role in defining its behavior, guiding the search and biasing the quality of the solutions found. However, adjusting the parameters is not a simple task, becoming an optimization problem within the problem being optimized. In addition, an appropriate setting for the parameters may change during the optimization process making this task even harder. There are two ways to adjust the parameters of an algorithm. The offline control that is performed before running the algorithm and parameter values remains fixed and the online control where parameter values may change during the optimization process. This article focuses on reviewing the online parameter control strategies applied in Evolutionary Computation and Swarm Intelligence. As a result, this review analyzes and points out the key techniques and algorithms used and suggests some directions for future research.

Key-words: Parameter Setup, Evolutionary Computation, Swarm Intelligence, Online Control

1 Introdução

A natureza oferece uma grande diversidade de fontes de inspiração para o desenvolvimento de algoritmos de otimização. Tais algoritmos são chamados de algoritmos inspirados na natureza e estão inseridos na grande área de pesquisa chamada Computação Natural. As metaheurísticas biologicamente inspiradas são reconhecidas como abordagens eficientes para resolução de diversos problemas difíceis de otimização [20]. Os algoritmos que compõem esta classe de estratégias de otimização podem utilizar diferentes rotinas de diversificação e intensificação na busca pela solução ótima [106]. A diversificação refere-se a exploração global, enquanto a intensificação refere-se a exploração local no espaço de busca das soluções do problema [17, 151]. Dentre os diferentes algoritmos bio-inspirados pode-se destacar dois grupos: a Inteligência de Enxame e a Computação Evolutiva.

A Inteligência de Enxame se caracteriza por algoritmos que possuem inspiração no comportamento coletivo de insetos, como formigas, abelhas, cupins e também de animais como peixes e pássaros [106]. Neste grupo encontram-se, por exemplo, o algoritmo de colônia de formigas (*Ant Colony Optimization*)[38] inspirado no comportamento de busca por

alimento das formigas, otimização de enxame de partículas (*Particle Swarm Optimization*) [64] inspirado no movimento de cardumes de peixes e bandos de pássaros, algoritmo da abelha (*Bee Algorithm*) [109] inspirado na busca de alimentos das abelhas, dentre outros [106].

A Computação Evolutiva é outro grupo importante dentro das metaheurísticas bio-inspiradas se caracterizando por algoritmos inspirados essencialmente pelas leis de Darwin, onde os indivíduos mais fortes e adaptados ao ambiente possuem maior chance de sobreviver e evoluir. Nesta analogia, os indivíduos representam soluções candidatas para otimizar dado problema e o ambiente representa o espaço de soluções. A Computação Evolutiva simula a evolução dos indivíduos através de processos de seleção, reprodução, *crossover*, e mutação, produzindo desta maneira soluções melhores. Alguns algoritmos que pertencem a este grupo são os algoritmos genéticos (*Genetic Algorithm*) [34], programação genética (*Genetic Programming*) [70], estratégia evolutiva (*Evolution Strategy*) [13], evolução diferencial (*Differential Evolution*) [133], dentre outros [20].

Apesar da variedade de algoritmos para otimização que os ambos grupos fornecem, estes compartilham, na sua maioria, algumas características em comum como: utilização de componentes estocásticos e variados parâmetros de configuração [151]. A quantidade de parâmetros a ajustar pode variar desde poucos como no algoritmo de vaga-lumes [151] e na otimização de colônias por abelhas artificiais [63] com 2 parâmetros cada, até muitos parâmetros como na otimização de enxame de partículas [64] e na otimização por colônia de bactérias [107] com 4 e 7 respectivamente.

Em qualquer que seja a metaheurística, o ajuste de seus parâmetros possui um papel importante, por controlarem a sua execução e influenciarem no direcionamento da busca por regiões promissoras no espaço de soluções do problema. De acordo com Eiben (1999) [41], a configuração dos parâmetros afeta diretamente a qualidade da solução final, sendo necessário ter conhecimento de qual deve ser a configuração mais promissora. Além do espaço de soluções do problema sendo abordado, passa a existir também o espaço de busca dos valores possíveis dos parâmetros de configuração do algoritmo sendo empregado. Surge assim um problema de otimização dentro do problema que está sendo otimizado. Eiben (1999) ainda destaca que há duas maneiras para ajustar parâmetros. A primeira conhecida como ajuste *offline* (*Parameter Tuning*), é realizada antes da execução do algoritmo onde diversos testes são executados previamente com diferentes configurações de parâmetros a fim de encontrar bons valores para os parâmetros. Com esta análise prévia objetiva-se obter valores padrão para os parâmetros que podem ser recomendados para execuções futuras. Porém, tais recomendações não devem ser generalizadas para todas as classes de problemas. A segunda maneira é conhecida como ajuste *online* ou controle *online* (*Parameter Control*), onde os valores dos parâmetros se alteram ao longo da execução. Vale ressaltar que o ajuste *online* elimina a etapa de análise prévia dos valores dos parâmetros de modo que o ajuste ocorre durante o processo de otimização. Consequentemente, no ajuste *online* o usuário ou projetista do algoritmo de

otimização se abstem da responsabilidade de ajustar tais parâmetros.

O ajuste *offline* tem se demonstrado um problema combinatorial complexo devido a alta quantidade de valores que cada parâmetro pode assumir [4]. Diversos trabalhos têm sido desenvolvidos para alcançar este tipo de ajuste [119, 37, 129, 42, 91]. Um trabalho de destaque nesta linha é realizado por Eiben (2011) [42], onde são revisados vários métodos de ajuste *offline*, uma taxonomia é proposta e é realizada a análise dos parâmetros de alguns métodos.

Já no controle *online*, uma característica importante é que o valor de um parâmetro pode variar de acordo com o estágio da busca [54]. Possuir controle *online* durante o processo de otimização pode melhorar significamente o resultado final e isto tem motivado a pesquisa por métodos para auto-ajustar os parâmetros inerentes às diferentes metaheurísticas [61, 71, 138, 5, 74, 127, 84]. O foco deste trabalho se concentra na revisão de técnicas para controle *online*.

Com o propósito de classificar os diferentes métodos de controle *online*, na literatura atual, encontram-se os trabalhos realizados por Angeline [9] apresentando uma proposta de taxonomia para o ajuste de controle, Smith e Forgaty [130] com uma revisão de operadores e ajuste de parâmetros em algoritmos genéticos, Eiben [41] com uma nova proposta de taxonomia de ajuste de parâmetros, De Jong [35] apresentando uma perspectiva histórica do ajuste de parâmetros, Zhang [160] com uma revisão de métodos de ajuste de parâmetros e proposta de uma taxonomia e Kramer [71] com uma revisão de métodos de ajuste de controles com destaque para o controle auto-adaptado. Destaca-se também o trabalho de Aleti [5] que realizou uma revisão de mecanismos de *feedback* para utilizar no ajuste da adaptação de parâmetros. Seguindo na linha de revisão de métodos no contexto de Inteligência de Enxame encontra-se o trabalho realizado por Rezaee (2013) [118] apresentando uma revisão de métodos para o ajuste de parâmetros.

Apesar de haver diversas propostas de classificação e revisão de métodos de ajuste de parâmetros, estas abordagens não foram aplicadas em ambas as áreas de Computação Evolutiva e Inteligência de Enxame. O presente artigo tem como propósito identificar as técnicas mais utilizadas, revisar e classificar os métodos aplicados especificamente no controle *online* de parâmetros, como também demonstrar as principais características e técnicas de cada uma. A presente revisão tem foco em trabalhos recentes envolvendo algoritmos da Computação Evolutiva e da Inteligência de Enxame no que tange a aplicação de técnicas para controle *online* de parâmetros. A revisão foi realizada nos principais mecanismos de buscas (*IEEE XPlorer, Science Direct, Scopus, Spring Link*, etc) com as palavras-chave *parameter control, adjust* e *adaptive*, combinadas com o nome de cada um dos algoritmos.

As próximas seções estão organizadas da seguinte maneira: A Seção 2 apresenta os algoritmos bio-inspirados abordados nesta revisão, especificamente da Computação Evolu-

tiva e Inteligência de Enxame; Seção 3 apresenta a fundamentação teórica e trabalhos relacionados ao estudo, como também a taxonomia a ser utilizada neste trabalho; A Seção 4 apresenta as técnicas de controle de parâmetros; A Seção 5 descreve as diferentes técnicas de ajuste de parâmetros aplicadas a algoritmos bio-inspirados; Na Seção 6 é discutida as importantes características das técnicas apresentadas e descreve a classificação proposta; Na Seção 7, as conclusões gerais e trabalhos futuros são relatados.

2 Algoritmos Bio-Inspirados

A partir da sua fonte de inspiração é possível separar os algoritmos bio-inspirados em diferentes grupos, como visto na Figura 1 [47].

Os algoritmos inspirados por sistemas físico-químicos são aqueles que foram projetados para reproduzir o comportamento e as características de certas leis da química ou da física, incluindo a gravidade, cargas elétricas, sistemas fluviais, entre outros. Neste grupo tem-se algoritmos como *Big Bang-Big Crunch Optimization* [43], otimização inspirado em buraco negro (*Black Hole Optimization*) [52], arrefecimento simulado (*Simulated Annealing*) [67], busca harmônica (*Harmony Search*) [48] dentre outros [47].

Os algoritmos baseados em sistemas biológicos, ou bio-inspirados são aqueles que são inspirados por características oriundas da Biologia. Nesta categoria encontram-se, por exemplo, as Redes Neurais Artificiais [49], os Sistemas Imunológicos Artificiais [30], a Computação Evolutiva [34] e a Inteligência de Enxame [106], sendo os dois últimos o foco deste trabalho dada a grande variedade de algoritmos de otimização.

A Computação evolutiva é baseada na lei da evolução natural de Darwin, onde indivíduos de uma população lutam pela sobrevivência e apenas os indivíduos mais fortes e adaptados têm a maior possibilidade de sobreviver e dar continuidade à sua espécie. Neste domínio encontram-se os Algoritmos Genéticos (*Genetic Algorithm*) [34], evolução diferencial (*Differential Evolution*) [133], estratégia evolutiva (*Evolucionary Strategy*) [13], algoritmos coevolutivos (*Coevolutionary Algorithms*) [20] e coevolutivos com inspiração ecológica (*Eco-inspired Evolutionary Algorithm*) [105].

Os algoritmos de Inteligência de Enxame são baseados no comportamento social e coletivo de insetos, como formigas, cupins, abelhas e agrupamentos de peixes ou aves. A inteligência destes sistemas encontra-se no comportamento coletivo dos indivíduos que é gerado através de pequenas interações individuais e é chamado de comportamento emergente Neste grupo encontram-se a otimização por colônia de formigas (*Ant Colony Optimization*) [38], otimização por colônia de abelhas artificiais (*Artificial Bee Colony*) [62], algoritmo de cardume artificial (*Artificial Fish School Algorithm*) [93], otimização por colônia de bactérias (*Bacterial Foraging Optimization*) [107], algoritmo do morcego (*Bat Algorithm*) [152], algo-

ritmo de vaga-lumes (*Firefly Algorithm*) [151], otimização por enxame de partículas (*Particle Swarm Optimization*) [65], dentre outros [72, 106].

Como visto na Figura 1 tem-se a distribuição dos grupos de algoritmos que são inspirados na natureza. O presente trabalho tem o objetivo de revisar os algoritmos bio-inspirados, especificamente os inseridos na Computação Evolutiva e na Inteligência de Enxame.

Figura 1. Conjunto dos algoritmos inspirados na natureza. Adaptado de Fister [47].

Dos parâmetros a serem analisados em cada algoritmo, esta revisão considera somente os parâmetros de entrada dos algoritmos em suas versões canônicas. Na sequência são descritos os principais algoritmos encontrados com suas especificações de parâmetros. Os 4 primeiros referem-se a algoritmos da Computação Evolutiva e os 8 seguintes são algoritmos da Inteligência de Enxame.

2.1 Algoritmos Genéticos

Os algoritmos genéticos (*Genetics Algorithms* - GA) são um dos algoritmos mais conhecidos e mais utilizados da Computação Evolutiva e foi proposto por John Holland em 1975 [33]. A grande inspiração que originou os GAs, é a evolução das espécies baseado na

teoria da evolução de Darwin. Na natureza, indivíduos de diferentes populações competem entre si para sobreviver. Segundo a seleção natural, os indivíduos mais fortes e mais bem adaptados ao ambiente têm maior chance de sobrevivência e de dar continuidade à sua espécie. Desta maneira os GAs utilizam os conceitos da evolução como um processo inteligente de otimização na busca de boas soluções.

O *crossover* simula a reprodução gerando filhos com a combinação de indivíduos previamente selecionados. Em cada indivíduo filho gerado é aplicado o operador de mutação. A mutação é responsável pela diversificação e o *crossover* pela intensificação da busca. Os parâmetros do GA são: o tamanho da população (*POP*), a probabilidade de *crossover* (*CR*), a probabilidade de mutação (*MUT*) e o tamanho do torneio (K) quando utilizado a seleção por torneio.

2.2 Evolução Diferencial

A evolução diferencial (*Differential Evolution* - DE) foi proposta por Storn e Price [133]. Os indivíduos ou vetores de solução são inicialmente gerados através de escolhas aleatórias. A criação de novos indivíduos é realizada através da adição da diferença ponderada entre dois indivíduos a um terceiro indivíduo, chamado de indivíduo alvo ou *target*. Esta operação é chamada de mutação, e cada um dos indivíduos que compõem a diferença são selecionados aleatoriamente e são mutuamente diferentes. O indivíduo gerado pela mutação é combinado com um outro indivíduo pré-selecionado aleatoriamente resultando no indivíduo teste (*trial*). Esta operação é chamada de *crossover*. Caso o indivíduo *trial* tiver seu valor de *fitness* pior em comparação ao indivíduo *target*, ele é descartado. Caso contrário, o indivíduo *trial* assume o lugar do indivíduo *target* na próxima geração e esta operação corresponde a seleção. Cada um dos indivíduos da população deverá ser o indivíduo *target* em uma geração. Este ciclo se repete até um critério de parada. Os parâmetros da DE são: o tamanho da população (*POP*), a probabilidade de *crossover* (*CR*) e o fator de mutação (F).

2.3 Estratégia Evolutiva

A estratégia evolutiva (*Evolution Strategy* - ES) foi proposta por Rechenberg e desenvolvida por Schwefel [16] nos anos 60. O primeiro algoritmo de ES possui um simples mecanismo de mutação e seleção e é chamado de ES com dois membros ou (1+1)-ES. Este modelo trabalha com um indivíduo, que gera um filho através de uma mutação a cada geração. Estes dois indivíduos são comparados em relação ao seu *fitness* e é eliminado o pior indivíduo. Este ciclo se repete até um determinado critério de parada. Outros modelos de ES com população de indivíduos foram propostos, como em (μ + λ)-ES e (μ , λ)-ES permitindo a utilização do operador de *crossover*. No primeiro modelo, μ indivíduos geram λ filhos, criando uma população temporária de (μ + λ) indivíduos. A partir desta população temporária

é realizada a seleção de μ indivíduos para a próxima geração. No modelo (μ , λ)-ES, tem-se $\mu < \lambda$, onde os μ indivíduos geram λ filhos e a população da próxima geração é formada apenas pelos indivíduos do conjunto λ. Outras versões do ES com população de indivíduos são os (μ/ρ+λ)-ES e (μ/ρ,λ)-ES. Nestas versões, o parâmetro ρ refere-se ao número de indivíduos envolvidos na geração de um novo indivíduo. Além dos parâmetros μ, λ e ρ citados acima, o ES possui o parâmetro δ que é a variância da mutação, chamado também de passo de mutação (*mutation step*).

2.4 Algoritmo de Inspiração Ecológica

O algoritmo de inspiração ecológica (*Ecological-inspired Algorithm* - ECO) foi proposto por Parpinelli [105] e apresenta uma perspectiva ecossistêmica para o desenvolvimento de algoritmos coevolutivos. O ECO é composto por populações de indivíduos, onde cada indivíduo representa uma solução potencial para o problema, e cada uma das populações evolui de acordo com uma estratégia de busca. Cada estratégia de busca utiliza os seus próprios parâmetros e mecânismos de intensificação e diversificação para modificar uma população de indivíduos. Os parâmetros do ECO são: o número de populações (*N-POP*), o tamanho da população (*POP-SIZE*), o número de ciclos de sucessões ecológicas (*ECO-STEP*), o tamanho do período evolutivo (*EVO-STEP*) e o limiar de proximidade para a definição dos habitats (ρ).

2.5 Algoritmo de Cardume Artificial

O algoritmo de cardume artificial (*Artificial Fish School Algorithm* - AFSA) foi proposto por Li [144] e tem como inspiração o comportamento dos cardumes dos peixes. Na natureza os peixes podem procurar lugares com uma quantidade maior de alimento de maneira individual ou seguindo outros peixes. Uma área que possui uma maior concentração de peixes, provavelmente é mais nutritiva. Seguindo esta regra, o AFSA utiliza peixes artificiais que procuram uma solução ótima no espaço de solução (o ambiente em que os peixes artificiais vivem), imitando o comportamento de cardumes dos peixes. Os parâmetros do AFSA são: o tamanho da população (*FishNum*), o número de iterações da busca (*Trynumber*), a influência da visão dos peixes (*Visual*) e o grau de congestionamento dos cardumes (δ).

2.6 Algoritmo do Morcego

O algoritmo do morcego (*Bat Algorithm* - BA) foi proposto por Yang [152]. O algoritmo é inspirado no processo de eco-localização dos morcegos, utilizado durante o seu voo para detectar presas e evitar obstáculos. A eco-localização se baseia na emissão das ondas ultrassônicas e correspondente medição do tempo gasto para estas ondas voltarem ao ponto de origem após serem refletidas pelo alvo ou obstáculo. A amplitude e o pulso dos sons emitidos pelos morcegos variam de acordo com a estratégia de caça. O controle da diversificação e

intensificação do algoritmo é realizado com a variação entre a amplitude e a taxa do pulso de cada morcego. Os parâmetros do BA são: o tamanho da população (n), o fator de decaimento da amplitude (α) e o fator de incremento da emissão de pulso (γ).

2.7 Algoritmo de Vaga-lumes

O algoritmo de vaga-lumes (*Firefly Algorithm* - FA) foi proposto por Yang [151]. A sua inspiração se encontra no padrão de luminosidade dos vaga-lumes da família *Lampyridae* (Ordem *Coleoptera*). A luz produzida pelo vaga-lume pode servir para atrair parceiros para o acasalamento, alertar do perigo de potenciais predadores e atrair presas. A partir da observação dos vaga-lumes o FA foi projetado utilizando três regras básicas: Os vaga-lumes são atraídos pelos outros, independente de seu sexo; A atratividade para o acasalamento é proporcional a intensidade de luz emitida e diminui à medida que a distância do vaga-lume observador aumenta; A intensidade da luz do vaga-lume é afetada ou determinada pela superfície da função objetivo do problema sendo otimizado. Os parâmetros do FA são: o tamanho da população (n) e o coeficiente de absorção (γ).

2.8 Otimização de Colônia por Abelhas Artificiais

A otimização por colônia de abelhas artificiais (*Artificial Bee Colony Algorithm* - ABC) é um algoritmo de inteligência de enxame e foi proposto por Karaboga [63]. O algoritmo consiste em reproduzir a busca de alimentos realizado pelas abelhas. Estas abelhas tem como o objetivo encontrar fontes de alimento (regiões do espaço de busca) com uma quantidade elevada de néctar. Cada abelha representa uma solução para o problema e consequentemente aponta para uma localidade no espaço de soluções. O ABC utiliza três tipos distintos de abelhas: as escoteiras (*scout bees*) que voam livremente pelo espaço de busca sem uma orientação específica; as abelhas empregadas (*employed bees*) que exploram a vizinhança de sua localização; e as abelhas expectadoras (*onlooker bees*) que selecionam probabilisticamente uma localidade para explorar a sua vizinhança. O ABC utiliza para intensificação da busca as abelhas empregadas e espectadoras. Para a diversificação utiliza as abelhas escoteiras. Os parâmetros do ABC são: o tamanho da população (*swarmsize*) e o número de tentativas das abelhas empregadas (*limit*).

2.9 Otimização por Colônia de Bactérias

A otimização por colônia de bactérias (*Bacterial Foraging Optimization* - BFO) foi proposta por Passino [107], inspirada no comportamento social de busca por alimento das bactérias *Escherichia coli* (E.coli). Durante o processo de busca a bactéria se move em pequenos passos enquanto procura alimento. Seu movimento é realizado através de um conjunto de filamentos conhecido como flagelos, que ajudam a bactéria se mover em períodos alterna-

dos de natação (*swim*) e tombos (*tumble*). A alternância entre estes dois períodos chama-se quimiotaxia. Cada bactéria representa uma solução para o problema. O ambiente fornece o substrato para as bactérias interagirem e é representado pelo espaço de busca sendo otimizado. Quanto melhor for a região do espaço de busca, melhor será o resultado da função objetivo, e consequentemente, melhor será o substrato para as bactérias. Os parâmetros do BFO são: o tamanho da população (*S*), o número de passos quimiotáticos (N_c), o número de etapas de caminhadas (N_s), o número de etapas de reprodução (N_{re}), o número de etapas de eliminação e dispersão (N_{ed}), a probabilidade de eliminação (p_{ed}) e o tamanho do passo de cada caminhada ou tombo ($C(i)$, para cada bactéria i).

2.10 Otimização por Colônia de Formigas

A otimização por colônia de formigas (*Ant Colony Optimization* - ACO) foi proposta por Dorigo [38]. O ACO é inspirado no comportamento de formigas reais na busca de alimentos, onde mesmo com visão debilitada, podem encontrar o caminho mais curto entre a fonte de alimento e sua colônia. A formiga em geral começa sua busca por alimento de forma aleatória e deposita no solo uma substância química chamada feromônio, formando uma trilha. Quando uma formiga encontra uma trilha de feromônio, ela é atraída e tem preferência em seguir as trilhas com maior quantidade desta substância. Por ser uma substância volátil, o feromônio evapora com o passar do tempo. Quanto mais formigas seguem um determinado caminho, mais difícil é do feromônio evaporar e mais atrativo se torna para as próximas formigas que porventura passarem por este caminho. Desta maneira, caminhos longos e poucos utilizados são abandonados levando a definição de caminhos mais curtos entre a fonte de alimento e a colônia. Os parâmetros do ACO são: o peso relativo da trilha (α), a atratividade da trilha (β), o tamanho da população (*m*) e a taxa de evaporação dos feromônios (ρ).

2.11 Otimização por Enxame de Partículas

A otimização por enxame de partículas (*Particle Swarm Optimization* - PSO) foi proposta por Kennedy (1995) [64, 65]. O PSO tem como inspiração o comportamento coordenado dos movimentos dos pássaros e cardumes de peixes. O algoritmo utiliza uma população de partículas que são geradas de forma aleatória no espaço de busca do problema. Cada partícula é uma solução potencial para o problema, representada por sua velocidade, localização no espaço de busca e uma memória que armazena a sua melhor posição visitada. O movimento de cada partícula depende de sua própria velocidade e da localização das boas soluções encontradas. Os parâmetros do PSO são: o tamanho da população (N), o peso da inércia (ω) e os coeficientes de aceleração (C_1 e C_2).

2.12 Algoritmo de Busca Gravitacional

O algoritmo de busca gravitacional (*Gravitational Search Algorithm* - GSA) foi proposto por Rashedi [115] e é uma releitura do algoritmo PSO. No GSA, cada objeto representa uma partícula e sua massa representa sua função objetivo. A principal diferença está na comunicação local entre os objetos que é determinada pelas massas. No GSA, cada objeto se torna um atrator. Cada solução potencial do problema é representada como sendo um objeto e sua qualidade é medida pelas suas massas. Os parâmetros do GSA são: a constante gravitacional (G_0), o parâmetro de controle da constante gravitacional (α) e o tamanho da população (*POP*). O parâmetro α do GSA controla o poder da diversificação e exploração.

3 Taxonomias para Controle de Parâmetros

Uma taxonomia considerando o controle de parâmetros de um algoritmo tem como objetivo classificar e organizar os métodos de acordo com características comuns. Como diferentes características podem ser utilizadas para definir tal classificação, diferentes taxonomias podem ser definidas. Na literatura são encontradas as taxonomias propostas por Angeline [9], Eiben [41] e Zhang [160], e são descritas a seguir.

Angeline [9] propôs uma taxonomia para controle de parâmetros baseada no tipo de regra de atualização e no nível onde é realizada a adaptação do parâmetro. A regra de atualização pode ser classificada como: absoluta ou empírica. As regras absolutas são predeterminadas e especificam como as modificações devem ser executadas. A regra empírica define uma função específica para o controle e permite que o próprio processo algorítmico determine se as mudanças nos parâmetros são vantajosas ou não. Esta regra pode ser chamada de auto-adaptada (ou agregada) quando os parâmetros são codificadas juntamente com a solução e o algoritmo evolui os seus respectivos valores.

Ainda segundo Angeline, a adaptação pode ser realizada em três níveis: população, indivíduo e componente. No nível da população, estão as técnicas que adaptam dinamicamente os parâmetros que são globais para toda a população. Como por exemplo, a probabilidade global de *crossover*. No nível de indivíduo estão as estratégias que associam diferentes valores de parâmetros para cada indivíduo. Por exemplo, a probabilidade de cada indivíduo sofrer mutação. No nível de componente, são associados valores de parâmetros para cada componente durante a evolução de um indivíduo, que determina como cada componente será modificado durante a reprodução. Este nível de adaptação se distingue por especificar como manipular um único componente independente dos demais. Como por exemplo, a variância da mutação gaussiana. Na Figura 2 é mostrado o esquema proposto por Angeline.

Na taxonomia proposta por Eiben [41], o ajuste de parâmetros se divide em duas categorias principais: o ajuste *offline* e o ajuste ou controle *online*. O ajuste *offline* é caracte-

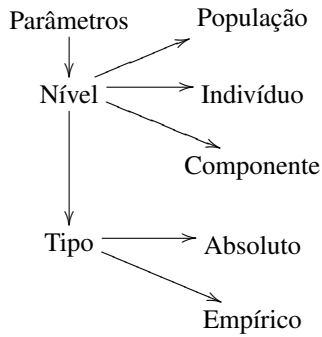

Figura 2. Taxonomia de Angeline

rizado por estratégias que utilizam valores de parâmetros estáticos e previamente ajustados. A categorização *offline* compreende o ajuste manual, ajuste por planejamento de experimentos e ajuste por meta-evolução [71]. O ajuste manual depende da experiência do usuário, onde o mesmo modifica os valores de parâmetros antes de cada execução do algoritmo. No planejamento de experimentos um conjunto de valores de parâmetros são experimentados previamente e através da análise dos resultados obtidos pela combinação dos valores dos parâmetros, é definida a melhor configuração a ser usada. A meta-evolução consiste em aplicar um algoritmo para evoluir os valores dos parâmetros.

O controle *online* executa a mudança do valor dos parâmetros durante a execução do algoritmo. O controle *online* é divido em três categorias: determinístico, adaptável e agregado. O controle determinístico é realizado através de regras determinísticas que alteram o valor do parâmetro sendo ajustado durante a execução, sem utilizar qualquer tipo de informação inerente ao processo de busca. O controle adaptável utiliza informações inerentes ao processo de busca para determinar a direção ou magnitude da mudança no valor do parâmetro sendo ajustado. O controle agregado realiza uma busca no espaço de valores dos parâmetros, onde os parâmetros são codificados juntamente com o vetor solução, se modificando durante o processo de otimização. Na Figura 3 é apresentada a visão geral da taxonomia proposta por Eiben.

A taxonomia proposta por Zhang [160] é constituída por três aspectos: o objeto da adaptação; a evidência da adaptação; e o método de adaptação. Esta taxonomia foi proposta considerando-se especificamente os algoritmos evolutivos. O objeto da adaptação caracteriza quais componentes do algoritmo se deseja adaptar e é dividido em quatro categorias: o controle de parâmetros (que é o foco deste trabalho), operadores evolucionários, estrutura da população e outros. A evidência da adaptação contém as informações que a adaptação pode se basear, como fatores determinísticos, valor de adaptabilidade, distribuição da po-

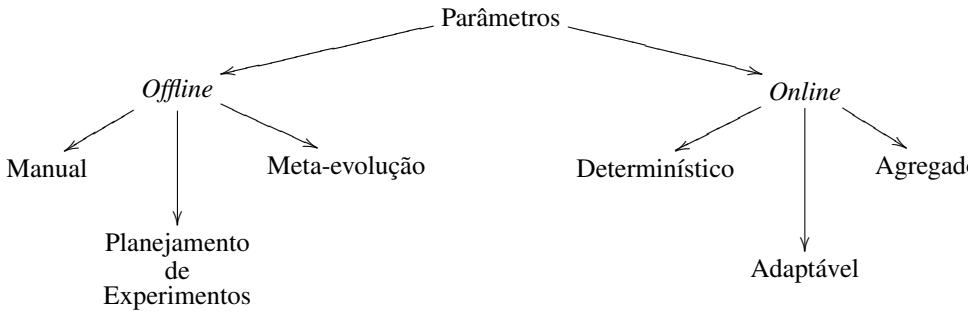

Figura 3. Taxonomia de Eiben (1999). Adaptado de [71]

pulação e a combinação do valor de adaptabilidade com a distribuição da população. O método de adaptação se refere ao mecanismo que a adaptação é realizada, podendo ser baseado em regras simples, na coevolução, no controle *fuzzy* ou no controle baseado em entropia. Na Figura 4 é apresentado a visão geral da taxonomia proposta por Zhang.

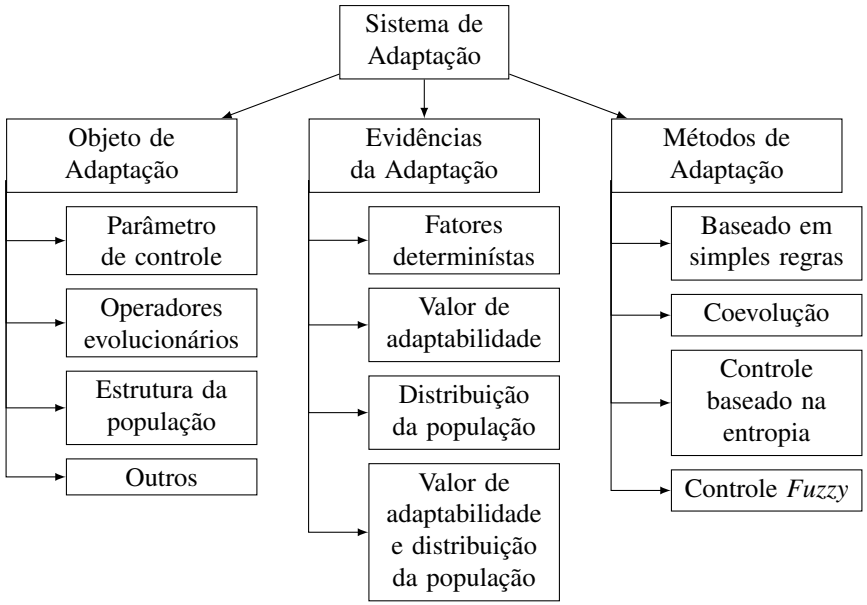

Figura 4. Taxonomia de Zhang (2012).

3.1 Considerações

A taxonomia de Angeline tem como principal meio de classificação o componente que pertence o parâmetro sendo ajustado como por exemplo, o operador de mutação. Ela não apresenta um modo de classificar as técnicas de controle pela característica mais importante que é o mecanismo de adaptação utilizada. A taxonomia proposta por Eiben (1999) permite ter uma visão global do ajuste de parâmetros de maneira simples. Claramente é possível identificar os diversos tipos de métodos utilizados no ajuste de parâmetros *offline* e *online* que são aplicados nos algoritmos bio-inspirados. Porém, esta taxonomia não apresenta uma maneira de classificar pelo mecanismo de adaptação. A proposta apresentada por Zhang permite a classificação pelo mecanismo de adaptação, além da classificação pelo componente e pela evidência de adaptação. Esta taxonomia pode ser vista como uma extensão da proposta feita por Eiben, onde o controle *online* de parâmetros é dividido em mais categorias, principalmente pelo mecanismo de adaptação. Como este trabalho tem foco no ajuste de parâmetros, a taxonomia de Eiben foi adotada porém com um melhor detalhamento no que se refere aos mecanismos de adaptação. Desta forma é possível identificar os métodos de ajuste de parâmetros aplicados em todas as metaheurísticas, sem se deter em características específicas de cada estratégia. Na Figura 5 é apresentada a taxonomia completa que será utilizada para classificar os tipos de ajuste de parâmetros neste trabalho.

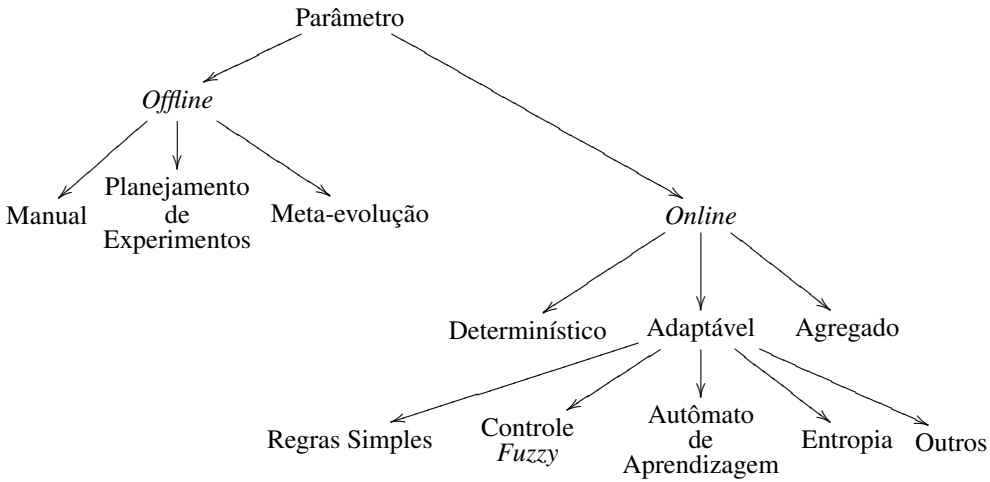

Figura 5. Taxonomia para Ajuste de Parâmetros. Adaptado de Eiben (1999) e Zhang (2012).

Na sequência são apresentados algumas técnicas de controle de seus parâmetros identificado na revisão.

4 Técnicas de Controle de Parâmetros

Esta seção descreve as principais técnicas de controle de parâmetros utilizadas pelos trabalhos encontrados durante a revisão da literatura. Sabe-se que a escolha da técnica correta de controle de parâmetros influencia diretamente no desempenho do algoritmo bio-inspirado sendo empregado [41]. Porém, independente da técnica de controle escolhida, tal uso se justifica apenas se houver melhoria no desempenho do algoritmo em comparação com o algoritmo original. Não há nenhuma regra que limite ou determine o uso de grupos específicos de técnicas. As técnicas encontradas durante a revisão da literatura para o controle de parâmetros variam, desde simples regras determinísticas, até métodos mais elaborados como *clustering* e autômatos de aprendizagem. As principais são apresentadas na sequência.

4.1 Determinístico

Este controle de parâmetros ocorre quando é aplicado uma regra determinística que altera o valor do parâmetro. A regra modifica o valor do parâmetro sem utilizar informações derivadas da otimização da função objetivo. Geralmente é utilizado um escalonamento de valores que varia com o tempo, por exemplo, o parâmetro começa com valores grandes e diminui gradualmente durante o processo de otimização. É comum a regra utilizar o número de gerações (iterações) ou número de avaliações de função objetivo para realizar o controle. Devido a simplicidade de implementação e o baixo custo computacional, várias aplicações utilizam esta forma de controle. Esta técnica de controle dificilmente captura o comportamento dos algoritmos bio-inspirados durante a otimização [160].

4.2 Agregado

Nesta técnica, os parâmetros são codificados diretamente no vetor solução, como dimensões extras, e são otimizados durante o processo de otimização da população. A otimização dos valores dos parâmetros pode ser realizada através das mesmas rotinas utilizadas para otimizar a população de soluções candidatas, como também pode ser realizada por rotinas específicas para os parâmetros [160]. Nesta abordagem, seguindo a analogia da seleção natural utilizada na Computação Evolutiva, os melhores valores dos parâmetros codificados que levam a melhores soluções, são mais propensos a propagar seus valores para gerações/iterações seguintes [41].

4.3 Regras Simples

Em muitas técnicas de controle de parâmetros, o controle é obtido através de regras simples que são definidas pela observação das características e do comportamento durante a execução do algoritmo [160]. As regras podem ser definidas através de funções lineares,

exponenciais, dentre outras. Este tipo de abordagem utiliza informações derivadas da otimização da função objetivo e modifica os valores dos parâmetros de acordo com o *feedback* recebido. O *feedback* pode ser o *fitness*, a diversidade da população, dentre outras.

4.4 Agrupamento

O agrupamento ou *clustering* é uma das tarefas fundamentais em mineração de dados e é utilizado para identificar padrões de maneira não supervisionada [57]. O objetivo do *clustering* é agrupar objetos de acordo com uma medida de similaridade [120]. A semelhança dos objetos de cada grupo pode ser medida através da distância, como por exemplo, a distância euclidiana [14]. Existem diferentes métodos de *clustering* como o hierárquico, particionado, baseado em densidade, baseado em modelos, baseado em *grids*, dentre outros [14]. O agrupamento pode ser utilizado para encontrar padrões entre os valores de parâmetros e o *feedback* obtido da otimização.

4.5 Autômatos de Aprendizagem

Autômatos de aprendizagem são mecânismos para tomada de decisão que podem operar em ambientes estocásticos e objetivam melhorar progressivamente o seu desempenho através de um processo de aprendizagem [98]. A aprendizagem é definida como sendo qualquer mudança relativamente permanente no comportamento do autômato e é resultante de experiências passadas. Desta maneira, a principal característica de um autômato de aprendizagem é sua capacidade de melhorar o seu desempenho com o passar do tempo. O autômato de aprendizagem pode ser utilizado para selecionar os valores de parâmetros através do seu processo de aprendizagem de acordo com os *feedbacks* obtidos da otimização.

4.6 Controle *Fuzzy*

O controle *fuzzy* é uma técnica de controle de parâmetros que trata os valores dos parâmetros utilizando graus de pertinência [160]. A lógica *fuzzy* ou lógica nebulosa, foi proposta por Zadeh em 1965 com o objetivo de inserir graus de incerteza na lógica tradicional booleana e permitir a inferência humana em conceitos e conhecimentos que não estão bem definidos. As partes básicas de um controlador *fuzzy* são a base de conhecimento, as regras *fuzzy* e o mecanismo de inferência [58]. No contexto de controle de parâmetros, o mecanismo de inferência é utilizado para avaliar o estado atual do sistema. As regras são propostas de acordo com o estado identificado e são utilizadas para controlar os valores dos parâmetros.

4.7 Entropia

O termo entropia foi proposto em Teoria da Informação por Shannon em 1948 [125]. A entropia pode ser definida como uma medida da informação esperada ou a incerteza de uma distribuição de probabilidades. Também é definida como o grau de desordem de um sistema ou a incerteza sobre uma partição [99]. Em algoritmos bio-inspirados, a geração de novas soluções candidatas frequentemente envolve fatores aleatórios e probabilísticos. Tal estocasticidade pode ser analisada com base na entropia da população de soluções candidatas. Portanto, a entropia pode ser utilizada para analisar as características da população de tal maneira que os parâmetros possam ser ajustados em conformidade com tal informação [160].

4.8 Matriz de Covariância

Em Teoria da Probabilidade a covariância é uma medida de dependência estatística entre duas variáveis aleatórias. As variáveis são independentes se possuem covariância zero. A matriz de covariância é uma matriz, simétrica, que sumariza a covariância entre diferentes variáveis. Esta técnica pode ser utilizada para aprender a dependência entre os parâmetros e aumentar a probabilidade de se repetir as etapas de sucesso [50].

4.9 Matriz de Feromônios

A matriz de feromônios é inspirada na substância química liberada pelas formigas [38]. Esta substância é depositada nos caminhos trilhados pelas formigas, possibilitando o reforço positivo. A quantidade depositada de feromônios é um atrativo para as formigas e a trilha com mais feromônios, provavelmente é a melhor. A matriz também utiliza o conceito de reforço negativo, em analogia a evaporação da substância. A matriz de feromônios pode ser utilizada para encontrar os melhores valores de parâmetros. Uma característica importante desta técnica é a cooperação, permitindo a utilização de inúmeros agentes para a construção da matriz.

5 Controle de Parâmetros em Algoritmos Bio-inspirados

Esta seção apresenta trabalhos encontrados na literatura que utilizam técnicas de ajuste *online* aplicadas em algoritmos bio-inspirados. A busca realizada cobre uma grande quantidade de trabalhos e enfatiza a aplicação do ajuste *online*. Esta revisão analisou as técnicas de controle, totalizando 162 parâmetros em um montante de 104 artigos. Com o objetivo de facilitar a identificação dos parâmetros envolvidos no controle, é apresentado na Tabela 1 um resumo dos algoritmos descritos na Seção 2 com seus respectivos parâmetros.

Da revisão, objetiva-se identificar quais parâmetros são ajustados e classificar as es-

	Algoritmo	Inspiração	Parâmetros
GA[33]	Algoritmo Genético	Teoria da Evolução	POP,CR,MUT,K
DE[133]	Evolução Diferencial	Teoria da Evolução	POP,CR,F
ES[16]	Estratégia Evolutiva	Teoria da Evolução	μ, λ, δ, ρ
ECO[105]	Algoritmo de Inspiração Ecológica	Ecologia	N-POP,POP-SIZE,ECO-STEP,EVO-STEP,ρ
AFSA[144]	Algoritmo de Cardume Artificial	Cardumes de peixes	Fishnum, Trynumber, Visual, δ
BA[152]	Algoritmo do Morcego	Eco-localização dos morcegos	n, α, γ
FA[151]	Algoritmo de Vaga-lumes	Padrão de luminosidade de vagas-lumes	n, γ
ABC[63]	Otimização de Colônia por Abelhas Artificiais	Busca por alimentos das abelhas	swarmsize,limit
BFO[107]	Otimização por Colônia de Bactérias	Busca por alimentos das bactérias *Escherichia coli*	S,N_c,N_s,N_{re},N_{ed},p_{ed}, C(i)
ACO[38]	Otimização por Colônia de Formigas	Busca por alimentos das formigas	α, β, ρ, m
PSO[64]	Otimização por Enxame de Partículas	Movimentos dos pássaros e cardumes de peixes	N,ω, C_1, C_2
GSA[115]	Algoritmo de Busca Gravitacional	Leis de gravidade e de movimento	G_0 α,, POP

Tabela 1. Algoritmos Bio-inspirados

tratégias utilizadas de acordo com a taxonomia vista na Seção 3. Com o objetivo de realizar a classificação, é utilizada uma nomenclatura específica que discrimina a técnica de controle utilizada e a quantidade de parâmetros ajustados por trabalho revisado. A Tabela 2 apresenta as siglas utilizadas. Por exemplo, a sigla TC-DT:2 representa uma técnica de controle determinístico que foi aplicada em 2 parâmetros diferentes.

Técnica de Controle	Sigla
Determinístico	TC-DT
Agregado	TC-AG
Regra Simples	TC-RS
Autômato de Aprendizagem	TC-AA
Controle *Fuzzy*	TC-CF
Clustering	TC-CL
Entropia	TC-ET
Matriz de Covariância	TC-MC
Matriz de Feromônios	TC-MF

Tabela 2. Sigla dos métodos de adaptação

5.1 Algoritmo Genético

Boeringer (2002) [18] apresenta em seu trabalho um método simples de controle da probabilidade da mutação. Através de uma função linear os limites máximo e mínimo são calculados e o valor da mutação é escolhido através de uma distribuição uniforme (TC-DT:1). Rongjun Li (2006) [77] apresenta uma função para adaptar as probabilidade de mutação e *crossover* que decrementa gradualmente os valores com relação ao número de gerações (TC-DT:2). Yuan-Yuan (2008) [36] apresenta um controle adaptável da probabilidade de *crossover* onde é considerado na adaptação do parâmetro a maturidade da população, medida através da média do *fitness* da população (TC-RS:1). Osaba (2013) [101] apresenta um GA com populações baseado em ilhas com método de controle para a probabilidade de *crossover*.

Para cada população o valor do parâmetro é ajustado com simples regras baseado no *fitness* do melhor indivíduo (TC-RS:1).

Vafaee (2009) [139] apresenta um modelo de controle onde a probabilidade de mutação é adaptada. O valor do parâmetro é adaptado de acordo com a frequência dos genes do melhor indivíduo utilizando modelos estatísticos da evolução de sequência de *DNA*. O modelo utilizado no trabalho é conhecido como modelo HKY (TC-RS:1). Linyu (2001) [149] apresenta um método de controle da probabilidade da mutação. O parâmetro é ajustado através de uma função linear com base na média do *fitness* da população (TC-RS:1). Yang (2013) [148] apresenta um método de controle para as probabilidades de *crossover* e mutação. O ajuste é realizado com uma função baseado no *fitness* do indivíduo e no grau de dispersão da população (TC-RS:2).

Aleti (2011) [3] propôs um método de previsão de parâmetros da mutação e *crossover*. O método consiste em utilizar a média do *fitness* da população juntamente com a taxa de sucesso do valor do parâmetro para ajustar a probabilidade da escolha dos valores em gerações futuras (TC-RS:2). Em outro trabalho, Aleti (2012) [6] propôs uma extensão do seu trabalho anterior. Nesta proposta os limites dos valores de parâmetro da mutação e *crossover* são ajustados conforme a evolução progride. A faixa de valores de melhor desempenho é reduzida pela metade, enquanto a faixa de pior desempenho é mesclada com suas faixas vizinhas (TC-RS:2). Aleti (2013) [4] apresenta mais uma extensão de seu trabalho. O controle consiste em utilizar *clustering* com algoritmo *k-means* baseado na qualidade dos valores dos parâmetros da mutação e *crossover*. Para cada grupo gerado pelo *clustering* é calculada a entropia que é utilizada para adaptar os valores dos parâmetros (TC-ET:2).

Boudjelaba (2014) [19] apresenta um GA híbrido. Em seu trabalho a probabilidade da mutação é ajustada para cada indivíduo com uma função com base no seu *fitness* e no *fitness* máximo e médio da população (TC-RS:1). Ponz-Tienda (2013) [111] apresenta um método de ajuste para a probabilidade de mutação e *crossover*. Os valores são ajustados de acordo com a quantidade de indivíduos inválidos que são gerados pelos operadores (TC-RS:2). Rajakumar (2013) [114] apresenta um método para controlar o tamanho da população. O ajuste é feito de acordo com o *fitness* da população e o número de gerações (TC-RS:1). Mary Linda (2013) [80] apresenta um método simples para ajustar a probabilidade de mutação. O valor é ajustado de acordo com o *fitness* dos indivíduos gerados (TC-RS:1). Chen [25] apresenta em seu trabalho uma função para ajustar a probabilidade de mutação e *crossover* baseado no *fitness* do indivíduo (TC-RS:2).

Aleti (2014) [2] apresenta um método para ajustar os parâmetros de um GA multiobjetivo e elitista. Os parâmetros adaptados são a probabilidade de mutação e a probabilidade de *crossover*. O método de controle consiste em utilizar a probabilidade Bayesiana para ajustar cada parâmetro com base em seu desempenho (TC-RS:2). Zhongliang (2013) [102] apresenta um método de controle para o tamanho da população e a probabilidade da mutação. O

tamanho da população é ajustado com uma função simples com base no *fitness* da população. A probabilidade de mutação é ajustado de acordo com o tamanho da população (TC-RS:2).

Yonggang (2014) [108] apresenta um GA híbrido com Arrefecimento Simulado (*Simulated Annealing*). Um controle *fuzzy* é utilizado para ajustar as probabilidades da mutação e *crossover* com base no *fitness* do indivíduo e da população (TC-CF:2). Tarokh (2014) [135] apresenta um método de controle *fuzzy* para as probabilidades da mutação e *crossover*. O ajuste é feito a cada geração de acordo com a medida de diversidade o *fitness* da população (TC-CF:2).

Smetek (2011) [128] realiza um estudo com dois modelos de ajuste. Cada modelo codifica uma quantidade diferente de parâmetros. O primeiro modelo de controle de parâmetros codifica no indivíduo os parâmetros da probabilidade de mutação e de *crossover* (TC-AG:2). No segundo modelo nenhum parâmetro é codificado nos indivíduo. A adaptação consiste em controlar o tamanho da população. A cada indivíduo é associado um valor inteiro que corresponde a sua idade e através da relação de seu *fitness* com a população este valor é atualizado. Quando a idade do indivíduo é menor ou igual a zero ele é retirado da população (TC-RS:1). Em seu estudo, o segundo modelo obteve melhores resultados.

Fernandez (2011) [45] apresenta um estudo com diferentes maneiras de adaptar a probabilidade de mutação. A primeira maneira proposta por Bäck (1996) [12], especifica uma função linear que diminui o valor da mutação de acordo com o número de gerações (TC-DT:1). A segunda abordagem apresentada neste trabalho consiste em utilizar valores aleatórios para o parâmetro de mutação a cada geração (TC-DT:1). A terceira abordagem é a codificação do parâmetro junto com o indivíduo, adicionando uma dimensão extra. Um operador específico é aplicado ao parâmetro, gerando um novo valor que é utilizado para aplicar a mutação no indivíduo (TC-AG:1). Na quarta abordagem, proposta por Srinivas (1994) [132], para cada indivíduo é associado um valor, que é calculado de acordo com o seu *fitness* (TC-RS:1). A quinta e última abordagem, aplica a lógica *fuzzy*, onde através da medida da convergência do *fitness* é realizado a adaptação com um conjunto de rótulos linguísticos (baixo, médio, alto) associado a diferentes valores de probabilidade de mutação (TC-CF:1). Segundo o estudo, a terceira abordagem, técnica agregada, obteve melhores resultados.

Um controle determinístico para várias abordagens do tamanho (K) do procedimento de seleção do tipo torneio é apresentado por Vajda (2008) [140]. O parâmetro é ajustado em função da geração, através de uma função linear (TC-DT:1). Um controle agregado também é apresentado, onde o valor do parâmetro K do torneio é codificado juntamente ao indivíduo em uma dimensão extra e $K \in [0, 1]$. O valor do K é correspondente a soma do valor da dimensão extra de todos os indivíduos da população (TC-AG:1). Um modelo híbrido de controle agregado é obtido através de uma função que ajusta o valor do parâmetro K de acordo com seu *fitness* se é pior ou melhor que seu pai (TC-RS:1). Um modelo de torneio *fuzzy* também é apresentado, onde através da diversidade genotípica e fenotípica é calculado

o parâmetro adaptado através de regras *fuzzy* (TC-CF:1). Segundo o estudo, o modelo híbrido do controle agregado obteve melhores resultados.

Como visto nos trabalhos apresentados, os parâmetros mais ajustados do GA são a probabilidade de mutação e *crossover*. Estes dois parâmetros são importantes por controlarem respectivamente a diversificação e intensificação da busca.

5.2 Evolução Diferencial

Huang (2013) [56] apresenta um método com função seno e cosseno para adaptar a probabilidade de *crossover* e o fator de mutação de forma determinista (TC-DT:2). Zhang (2013) [159] apresenta uma estratégia para adaptar a probabilidade de *crossover*. A estratégia utiliza o número de iterações para ajustar o valor do parâmetro (TC-DT:1). Wang (2013) [143] apresenta um DE com redimensionamento da população. O redimensionamento é definido por uma regras simples baseada na evolução do *fitness* do melhor indivíduo. Caso o melhor indivíduo não melhore, a população é reduzida, removendo indivíduos ruins da população. Caso o melhor indivíduo seja melhorado entre as iterações, a população aumenta gradualmente (TC-RS:1).

Brest (2006) [21] apresenta uma estratégia que controla os parâmetros de *crossover* e o fator de mutação. Estes parâmetros são codificados com os indíviduos e através de regras simples estes são ajustados (TC-AG:2). Teo (2006) [136] apresenta dois modelos de controle agregado para ajustar o parâmetro do tamanho da população de indivíduos. Nos dois modelos também é aplicado o controle agregado para os parâmetros de probabilidade de mutação e *crossover*. No primeiro modelo, o parâmetro do tamanho da população é codificado junto com o indivíduo (TC-AG:3). No segundo modelo, a taxa de crescimento da população é codificado no indivíduo, sendo um valor positivo para incrementar ou negativo para decrementar (TC-AG:3). Em seu estudo, o primeiro modelo se mostrou melhor por providenciar mais estabilidade para a otimização. Dragoi (2013) [39] apresenta um controle agregado para os parâmetros de *crossover* e o fator de mutação . Os valores são codificados e modificados com os mesmos operadores aplicados nos indivíduos (TC-AG:2).

Kovačević (2014) [69] apresenta um método de controle para o fator de mutação. É definido um conjunto de valores para o parâmetro e as respectivas probabilidades de cada valor são ajustados de acordo com sua taxa de sucesso (TC-RS:1). Piotrowski (2013) [110] apresenta um controle para o parâmetro de *crossover*. O valor do parâmetro é ajustado de acordo com seu sucesso em gerar soluções melhores (TC-RS:1). Zhong (2014) [161] apresenta um método baseado no *fitness* para adaptar a probabilidade de mutação (TC-RS:1). Ali (2004) [8] propôs uma regra adaptada baseada no *fitness* mínimo e máximo dos indivíduos para controlar o fator de mutação (TC-RS:1).

Chen (2014) [27] apresenta uma estratégia para ajustar o tamanho da população. O

ajuste é realizado com simples regras com base no *fitness* da população (TC-RS:1). Tanabe (2013) [134] apresenta um método para ajustar os parâmetros de *crossover* e fator de mutação F. O ajuste é realizado com base no histórico dos valores dos *fitness* (TC-RS:2). Mallipeddi (2010) [83] propôs um conjunto de estratégias de mutação e *crossover* associado com um conjunto de valores para cada parâmetro. Cada indivíduo da população inicial é atribuído aleatoriamente a uma estratégia de mutação e *crossover* associado com seus respectivos valores de parâmetros. Caso essa combinação produza um indivíduo melhor ela é armazenada, caso contrário ela é aleatoriamente reiniciada com uma nova estratégia e valores de parâmetros ou a partir das combinações de sucessos armazenadas com a mesma probabilidade (TC-RS:2). Liu (2005) [81] apresenta uma Evolução Diferencial com adaptação no controle dos parâmetro *crossover* e fator da mutação utilizando lógica *fuzzy*, tendo como entradas a diferença dos valores da função objetivo e os indivíduos da população (TC-CF:2). Kotinis (2014) [68] apresenta um controle *fuzzy* para os parâmetros de *crossover* e fator de mutação. O ajuste dos parâmetros é realizado com base na quantidade de soluções não dominadas e na diferença da distribuição da população entre duas iterações (TC-CF:2).

Como visto nos trabalhos apresentados acima, os parâmetros mais ajustados do DE são a probabilidade de *crossover* e o fator de mutação.

5.3 Estratégia Evolutiva

Uma estratégia de controle de parâmetros muito conhecida é a regra de 1/5 de Rechenberg [117]. Esta regra estabelece que a proporção de mutações bem sucedidas para todas as mutações devem ser de 1/5. Por conseguinte, se a razão for maior do que 1/5, o tamanho do passo de mutação deve ser aumentado e, se a proporção for inferior a 1/5, o tamanho do passo de mutação deve ser diminuído (TC-RS:1). Beyer (1995) [15] apresenta uma estratégia de controle de parâmetro agregado para a variância da mutação. O parâmetro é atualizado de acordo com funções específicas com base no *fitness* do indivíduo (TC-AG:1). O mecanismo fundamental do CMA-ES é a adaptação da matriz de covariância (CMA). Este mecanismo ajusta o tamanho do passo da mutação. A CMA é baseada na observação de que o modelo de distribuição deve ser atualizado de uma maneira acumulada. A informação do caminho de evolução também é usado para adaptar a matriz de covariância [160] (TC-MC:1).

5.4 Algoritmo de Inspiração Ecológica

Parpinelli (2012) [104] propôs um método para adaptar parâmetro de proximidade na definição dos habitats do *framework* ecológico. O método realiza um agrupamento hierárquico utilizando o algoritmo *single-link*. As informações de distância geradas pelo algoritmo *single-link* são utilizadas como probabilidades para conduzir a formação dos habitats (TC-CL:1).

5.5 Algoritmo de Cardume Artificial

Hongrui (2009) [55] apresenta um método de controle para o parâmetro δ com uma função determinista (TC-DT:1). Yazdani (2011) [154] apresenta um método de controle determinista para o controle do parâmetro *Visual* com componente aleatório em função do número de iterações (TC-DT:1). Yong (2011) [157] apresenta um método de controle determinista para o parâmetro *Visual*. O parâmetro é controlado a partir de uma variável α que é ajustada de acordo com uma função exponencial que considera o número de iterações (TC-DT:1). He (2009) [53] apresenta dois métodos deterministas para a adaptação do parâmetro *Visual*. O primeiro método é baseado no número de iterações e decrementa gradualmente até atingir o valor mínimo para o parâmetro (TC-DT:1). O segundo método também é baseado no número de iterações com a adição de constante β para decrementar gradualmente o valor do parâmetro (TC-DT:1).

Jiye (2013) [60] propôs também um método adaptável para controlar o parâmetro *Visual*. O ajuste é realizado com uma regra simples baseando-se no *fitness* do indivíduo e da média da população (TC-RS:1). Tian (2009) [137] apresenta um método que utiliza o valor do *fitness* para adaptar o parâmetro *Visual*. A função proposta decrementa o valor do parâmetro ao longo das iterações (TC-RS:1). Yazdani (2011) [155] propôs dois modelos *fuzzy* para controlar o parâmetro *Visual*. O primeiro modelo proposto chamado de *Fuzzy Uniform Fish*, utiliza como entrada para o controle *fuzzy* o número da iteração e a taxa de indivíduos com o melhor *fitness* (TC-CF:1). O segundo modelo chamado de *Fuzzy Autonomous Fish* utiliza como entrada para o controle *fuzzy* a distância do melhor indivíduo, o *ranking* de *fitness* e o número da iteração (TC-CF:1).

Como visto nos trabalhos apresentados acima, o parâmetro mais ajustado do AFSA é o *Visual*.

5.6 Algoritmo do Morcego

Yilmaz (2014) [156] apresenta um método determinista para o controle do parâmetro de amplitude (α) e emissão de pulso (γ). Para cada dimensão do indivíduo é atribuído um valor de parâmetro de amplitude e emissão de pulso. Os valores são ajustados com simples funções com base no número de iterações (TC-DT:2). Chen (2013) [28] propôs um método de adaptação do parâmetro de amplitude (α) e emissão de pulso (γ). A amplitude é ajustada com uma função simples com base no número de iterações (TC-DT:1). A emissão de pulso é ajustado com uma função simples com base no valor da amplitude (TC-DT:1).

Como visto nos trabalhos apresentados acima, os parâmetros ajustados do BA são a amplitude (α) e emissão de pulso (γ).

5.7 Algoritmo de Vaga-lumes

Abshouri (2011) [1] apresenta o FA combinado com um autômato de aprendizagem. O autômato é responsável pelo parâmetro de coeficiente de absorção (γ) e utiliza apenas o número de iterações como valor de entrada (TC-DT:1). Yan (2012) [146] apresenta um FA com o parâmetro de coeficiente de absorção (γ) adaptado. O parâmetro γ é controlado deterministicamente com decremento de valor, baseado na quantidade de dimensões do problema (TC-DT:1).

Fister (2013) [46] apresenta um FA com controle agregado de parâmetro. Cada indivíduo é codificado com o seu respectivo valor de coeficiente de absorção de luz (γ). Para o parâmetro é definida uma estratégia específica de mutação (TC-AG:1). Nikman (2012) [96] propôs um modelo agregado para o controle do parâmetro γ. O valor do parâmetro é codificado junto com o indivíduo (TC-AG:1). Selvarasu (2013) [124] apresenta um controle agregado para o parâmetro γ. Para ajustar o parâmetro, é utilizado os mesmos operadores aplicados nas soluções (TC-AG:1).

Roy (2013) [121] apresenta uma estratégia para controlar o parâmetro γ. Um conjunto de valores é utilizado e as respectivas probabilidades de cada parâmetro são ajustadas com base em suas performances em gerar novos indivíduos (TC-RS:1).

Como visto nos trabalhos apresentados acima, o parâmetro ajustado do FA encontrado nesta revisão é o coeficiente de absorção γ.

5.8 Otimização por Colônia de Abelhas Artificiais

Aydin (2014) [11] apresenta um método para ajustar o tamanho da população. A probabilidade de adicionar ou remover indivíduos da população é controlada por uma variável que é ajustada de acordo com a taxa de sucesso do algoritmo (TC-RS:1).

5.9 Otimização por Colônia de Bactérias

Dasgupta (2009) [31] apresenta um método de controle do parâmetro de tamanho do passo C. Os valores do parâmetro são adaptados através de uma função determinista (TC-DT:1). Chen (2011) [26] apresenta dois métodos de controle do parâmetro tamanho do passo C. No primeiro método, os valores dos parâmetros são atualizados a cada período, decaindo quando o *fitness* do melhor indivíduo melhora e se mantendo caso contrário. O valor do parâmetro é reiniciado no final de um período de t iterações (TC-RS:1). No segundo modelo, é utilizado um controle agregado para o parâmetro C que é ajustado com um operador específico para o parâmetro (TC-AG:1). Em seu estudo o primeiro modelo obteve resultados melhores. Farhat (2010)[44] apresenta um método determinista para decaimento linear do valor do parâmetro C (TC-DT:1). Panigrahi (2009) [103] propôs uma função simples para

controlar de modo determinista o parâmetro C (TC-DT:1). Rashtchi (2009) [116] apresenta um método para controlar o parâmetro C, onde este é ajustado de forma determinista com funções exponenciais (TC-DT:1). Yan (2012) [147] propôs um método de controle baseado no número de avaliações de indivíduos. O método é determinista e gera um decaimento gradual no parâmetro C (TC-DT:1).

Xin (2014) [145] apresenta uma função com base no *fitness* para ajustar o parâmetro C (TC-RS:1). Majhi (2009) [82] apresenta uma função para o caimento gradual do parâmetro C baseado no *fitness* do indivíduo (TC-RS:1). Datta (2008) [32] apresenta um método de controle para o parâmetro C, onde este é ajustado para cada indivíduo com base no seu *fitness* (TC-RS:1). Sathya (2011) [123] propôs um método baseado no *fitness* do indivíduo para controlar os valores do parâmetro C (TC-RS:1). Mezura (2012) [88] propôs um método para controlar o parâmetro C. O método consiste em ajustar o parâmetro de acordo com sua taxa de sucesso. (TC-RS:1). Mishra (2005) [89] propôs um controle *fuzzy* para adaptar o parâmetro C com base no *fitness* do melhor indivíduo da população (TC-CF:1). Venkaiah (2011) [141] propôs um método com controle *fuzzy* para o parâmetro C. O controle tem como variáveis de entrada o erro da função objetivo e valor atual de C, tendo como saída o valor ajustado para o parâmetro C (TC-CF:1).

Como visto nos trabalhos apresentados acima, o único parâmetro ajustado do BFO encontrado nesta revisão é o tamanho do passo C devido a sua importância no algoritmo.

5.10 Otimização por Colônia de Formigas

Merkle (2002) [86] propôs dois métodos de controle determinista para os parâmetros β e ρ, que representam a atratividade da trilha de feromônios e a taxa de evaporação, respectivamente. Para o parâmetro β o valor é decrementado em função da iteração enquanto o parâmetro ρ aumenta gradualmente em função da iteração (TC-DT:2). Meyer (2004) [87] apresenta um controle determinista para controlar o parâmetro α. A técnica consiste em utilizar uma função chamada α-*anneling*, definida pelo autor (TC-DT:1).

Cai (2008) [22] propôs um método onde para cada indivíduo é associado um valor para o parâmetro ρ. A adaptação do parâmetro é baseado na qualidade da solução gerada (TC-RS:1). Chusanapiputt (2006) [29] propôs um ACO com dois grupos distintos de população, onde os indivíduos migram de uma população para outra. Os parâmetros α e β são ajustados de acordo com o tamanho da população (TC-RS:2). Li (2009) [78] propôs um método de controle baseado em entropia. O método consiste em calcular a média da entropia e através de regra simples os valores dos parâmetros α e β são ajustados (TC-ET:2). Li (2013) [76] segue a mesma ideia do trabalho anterior, porém apresenta uma regra diferente para adaptar os valores dos parâmetros α e β (TC-ET:2). Castillo (2013) [23] apresenta um controle *fuzzy* para os parâmetros de taxa de evaporação ρ. O controle utiliza uma medida de erro para

ajustar os valores dos parâmetros (TC-CF:2). Khichane (2009) [66] propôs um controle para os parâmetros α e β, baseado em matriz de feromônio. A adaptação dos parâmetros ocorre em ciclos de iterações (TC-MF:2).

Como visto nos trabalhos apresentados acima, os parâmetros mais ajustado do ACO são o α e β.

5.11 Otimização por Enxame de Partículas

Yasuda (2010) [153] apresenta um mecanismo simples de adaptar, através de função lineares, os parâmetros ω, $C1$ e $C2$ (TC-DT:3). Niknam (2010) [94] utiliza uma função linear para adaptar o parâmetro ω (TC-DT:1). Shi (1999) [126] apresenta um método determinista com decremento linear do peso de inércia das partículas em função do número de iterações (TC-DT:1). Jiao (2008) [59] apresenta um método determinista com função não linear para adaptar o peso de inércia (TC-DT:1). Seguindo a mesma ideia de Jiao (2008) com diferentes funções não lineares tem-se os trabalhos de [7] e [24] (TC-DT:2). Uma função simples e aleatória para adaptar o peso de inércia é proposto por Eberhart (2001) [40] (TC-DT:1). Jin Wang (2011) [142] ajusta os parâmetros ω, $C1$ e $C2$. O parâmetro ω é controlado através de decremento linear em função do número de iterações (TC-DT:1), $C1$ através da utilização de informação da melhor partícula (*pbest*) e da melhor global (gbest) (TC-RS:1), o parâmetro $C2$ é incrementado através de uma função linear com base no número de iterações (TC-DT:1). Yang (2007) [150] apresenta uma abordagem para ajustar o parâmetro de peso da inércia ω. O parâmetro é ajustado com uma função baseado no *fitness* do melhor indivíduo e da média da população (TC-RS:1). Zielinski (2007) [162] apresenta um método de controle para os parâmetros ω, $C1$ e $C2$. Para cada geração é utilizado diferentes combinações de valores para os parâmetros e ajustado de acordo com o seu sucesso em gerar soluções melhores (TC-RS:3).

Hashemi (2011) [51] utiliza autômatos de aprendizagem (*learning automata*) para o ajuste dos parâmetros ω, $C1$ e $C2$. São apresentadas duas abordagens no uso dos autômatos de aprendizagem. Na primeira, chamada de aventureira, o autômato tem a liberdade de mudar drasticamente o valor do parâmetro. Na segunda, chamada de conservativa a mudança do valor é gradual, limitada por um valor fixo (TC-AA:3). Li (2009) [75] apresenta uma estratégia de aprendizagem para a seleção do operador de mutação. É apresentado quatro tipos diferentes de mutação que são iniciados com a mesma probabilidade. A partir da taxa de sucesso de cada operador a sua respectiva probabilidade é adaptada, respeitando o limite mínimo de probabilidade (TC-RS:1). Em Montalvo (2010)[90] é realizado o ajuste do parâmetro ω de maneira agregada ao indivíduo (TC-AG:1). Melin (2013) [85] apresenta um controle *fuzzy* para os parâmetros $C1$ e $C2$. O controle utiliza o número de iterações, uma medida de diversidade e uma medida de erro baseado no *fitness* para ajustar os valores dos parâmetros (TC-CF:2). Niknam (2012) [95] apresenta um controle *fuzzy* para os parâmetros

ω, $C1$ e $C2$. O controle utiliza o melhor valor de *fitness* e número de iterações que o *fitness* não melhorou para ajustar os valores dos parâmetros (TC-CF:3). Olivas (2013) [100] apresenta um controle *fuzzy* para os parâmetros $C1$ e $C2$. O controle utiliza uma medida de diversidade da população e número de iterações para ajustar os parâmetros (TC-CF:2). Neste mesmo trabalho é apresentado uma modificação do controle *fuzzy* onde as variáveis de entrada, a medida de diversidade e número de gerações são consideradas *fuzzy* (TC-CF:2). Em seu estudo o segundo modelo obteve resultados melhores. Neshat (2013) [92] apresenta um controle *fuzzy* para os parâmetros $C1$ e $C2$. O controle utiliza os valores de *fitness* da população para ajustar os parâmetros (TC-CF:2). Xiaolei (2014) [79] apresenta um controle para o parâmetro ω. O ajuste é realizado através da diferença entre os *cluster* da população e o melhor indivíduo (TC-CL:1).

Como visto nos trabalhos apresentados acima, os parâmetros mais ajustado do PSO são o peso da inércia ω, e os coeficientes de aceleração $C1$ e $C2$.

5.12 Algoritmo de Busca Gravitacional

Sombra (2013) [131] propôs um método de controle com lógica *fuzzy*, baseado na quantidade de iterações para controlar o parâmetro α. Esta estratégia não utiliza informações derivadas da evolução da busca (TC-DT:1). Precup (2012) [112] apresenta um método determinista com função exponencial para o escalonamento de valores para a constante gravitacional G (TC-DT:1). Em outro trabalho, Precup (2013) [113] apresenta um controle *fuzzy* para adaptar a constante gravitacional G. O controle utiliza apenas o número de iterações como informações de entrada para ajustar o parâmetro e nenhuma informação derivada da otimização (TC-DT:1). Nikman (2013) [97] propôs um controle adaptável para a constante gravitacional G. Em cada iteração o valor de G é gerado aleatoriamente respeitando os limites máximos e mínimos (TC-DT:1).

Zahiri (2012) [158] apresenta um método de controle *fuzzy* para o parâmetro da constante gravitacional. O controle utiliza o *fitness* do melhor indivíduo, o número de iterações que o melhor indivíduo não alterou, e a variância do *fitness* da população para ajustar os parâmetros. (TC-CF:1) Askari (2012) [10] propôs um GSA com controle *fuzzy* para ajustar a constante gravitacional G. O controle *fuzzy* tem como entrada o melhor *fitness* encontrado na iteração T, o número de iterações que o melhor *fitness* não se modificou e a variância dos *fitness* na iteração T. Com essas informações é ajustada a constante gravitacional (TC-CF:1). Kumar (2013) [73] apresenta o GSA com controle do parâmetro com lógica *fuzzy*. Em sua proposta é utilizado o valor do *fitness* do indivíduo normalizado e atual valor da constante gravitacional G e possui como saída do controle *fuzzy* o valor adaptado para a constante gravitacional (TC-CF:1). Saeidi-Khabisi (2012) [122] propôs um método utilizando lógica *fuzzy* para controlar o parâmetro α. Através da medida da diversidade da população, a medida de progresso da população e número de iterações é aplicado a lógica *fuzzy* para incrementar ou

diminuir o valor do parâmetro α (TC-CF:1).

Como visto nos trabalhos apresentados acima, os parâmetros mais ajustado do GSA são a constante gravitacional G e α.

6 Discussão

Os algoritmos bio-inspirados apresentados na Seção 2 possuem diversos parâmetros que precisam ser ajustados. Contudo, esta não é uma tarefa simples e com objetivo de encontrar a melhor forma de ajustar tais parâmetros, foram revisadas diferentes técnicas de controle (Seção 4). Suas aplicações nos algoritmos bio-inspirados foram apresentadas na Seção 5.

Na revisão da literatura apresentada neste trabalho, foram analisados diversos artigos científicos. Esta revisão analisou as técnicas de controle aplicadas em um montante de 104 artigos, totalizando 162 parâmetros .

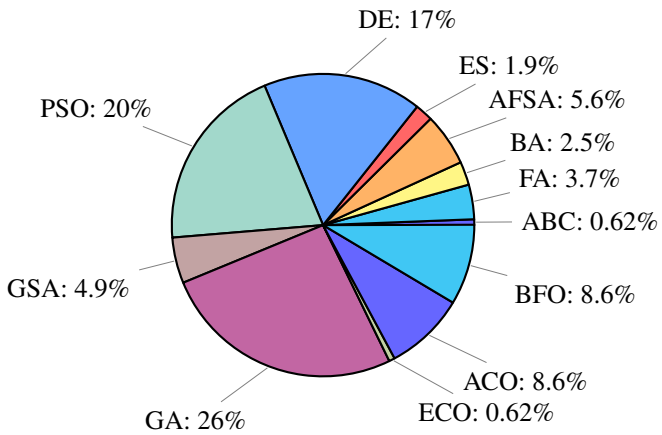

Figura 6. Distribuição dos algoritmos bio-inspirados em relação a quantidade de parâmetros ajustados

A Figura 6 apresenta a distribuição dos algoritmos bio-inspirados em relação a quantidade de parâmetros ajustados. Pode-se observar que rotinas para o controle de parâmetros são mais frequentemente aplicadas aos algoritmos genéticos, representando 26% do total dos métodos de controle analisados. A Otimização por Enxame de Partículas encontra-se em segundo com 20% e a Evolução Diferencial com 17% em terceiro. Esta distribuição pode ser explicada pelo fato que o GA, PSO e DE estarem entre os algoritmos bio-inspirados mais conhecidos, além de suas comprovadas eficiências em diferentes domínios. Também, a sim-

plicidade de implementação destes algoritmos pode ter contribuído significativamente. Deste extrato, verifica-se também a oportunidade de aplicação de técnicas de controle *online* nos demais paradigmas de otimização.

A Figura 7 sumariza a distribuição dos métodos de controle de parâmetros (determinístico, agregado e adaptativo) aplicados nos algoritmos bio-inspirados. O método mais utilizado é o controle adaptativo em 61% do total. Isto demonstra a preocupação da comunidade científica em utilizar informações de *feedback* do processo de otimização para adaptar os valores dos parâmetros, sabendo-se que dependendo do estágio da busca, os valores ótimos para os parâmetros podem ser diferentes. O controle determinístico vem em seguida com 27% dos métodos. A grande motivação em utilizar o controle determinista está na sua simplicidade de implementação, mesmo sem utilizar *feedback* durante o processo de otimização, os resultados obtidos com este controle tem se mostrado superior em comparação com o uso de valores fixos. O controle agregado foi encontrado em 12% dos trabalhos analisados. Pode-se concluir pela baixa utilização desta técnica que a aplicação dos mecânismos de otimização dos algoritmos para ajustar os parâmetros não é atrativa, possivelmente devido a um pequeno ganho na eficiência do algoritmo.

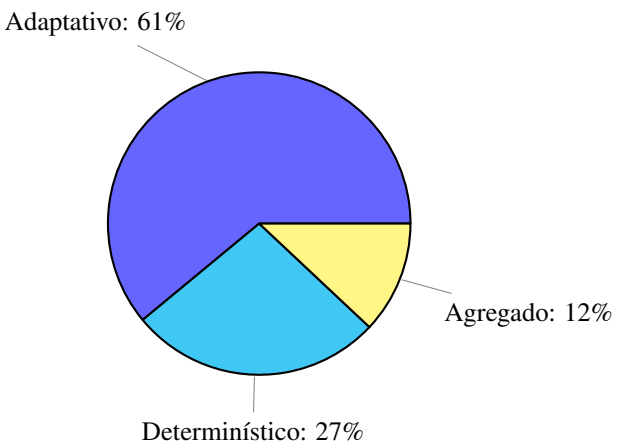

Figura 7. Distribuição dos mecanismo de controle

A Figura 8 mostra a distribuição dos métodos de controle por algoritmo bio-inspirado. Como reflexo da Figura 7, nota-se que para cada algoritmo, com exceção do BA, existe pelo menos um método de controle adaptativo e na maioria dos algoritmos existe um método determinista, com exceções de ES, ECO e ABC. Os métodos de controle agregado encontram-se principalmente nos algoritmos DE e GA, seguido pelos algoritmos de FA, BFO, PSO e ES.

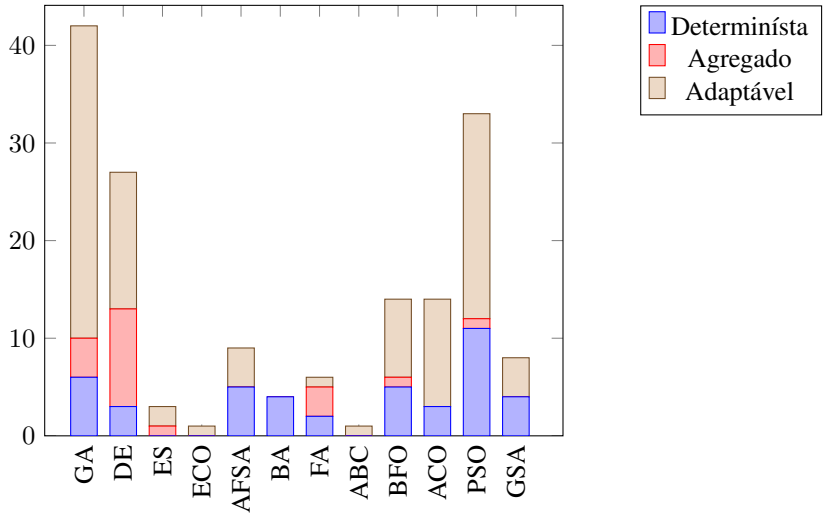

Figura 8. Distribuição dos mecânismos de controle por algoritmo

Para obter uma visão mais adequada da distribuição das técnicas de controle adaptativa é apresentada a Figura 9. Como visto na Seção 3, os métodos adaptativos são aqueles que utilizam alguma informação de *feedback* da otimização para ajustar os parâmetros. Neste grupo, encontra-se as técnicas de regras simples (TC-RS), autômatos de aprendizagem (TC-AA), controle *fuzzy* (TC-CF), *clustering* (TC-CL), entropia (TC-ET), matriz de covariância (TC-MC) e matriz de feromônios (TC-MF). A técnica mais comum é a regras simples, devido a sua simplicidade de implementação, representando 54%. Em seguida encontra-se o controle *fuzzy* com 32% do total. O controle *fuzzy* têm se demonstrado promissor, porém determinar as regras *fuzzy* e seus respectivos limiares pode ser uma tarefa difícil, principalmente se aplicado para ajustar muitos parâmetros. Com 2% encontra-se a matriz de feromônios. Esta técnica é interessante por poder ser utilizada para considerar a dependência entre os parâmetros sendo ajustados. A técnica de entropia possui 6.1%, seguido pelas técnicas de autômatos de aprendizagem, *clusters* e a matriz de covariância, respectivamente com 3.1%, 2% e 1%.

A Figura 10 sumariza as técnicas de controle adaptativo aplicadas aos algoritmos bio-inspirados. O algoritmo de ACO e PSO possuem a maior diversidade de controle adaptativo, com 4 técnicas diferentes cada, seguido por GA com 3 técnicas. Com exceção do algoritmo ECO, BA e GSA, todos os algoritmos possuem pelo menos uma técnica de regra simples. O controle *fuzzy* pode ser encontrado nos algoritmos de GA, DE, AFSA, BFO, ACO e GSA. O autômato de aprendizagem encontra-se apenas no PSO, o mesmo se repetindo para a técnica de matriz de feromônios que encontra-se apenas para o ACO e a matriz de covariância apenas

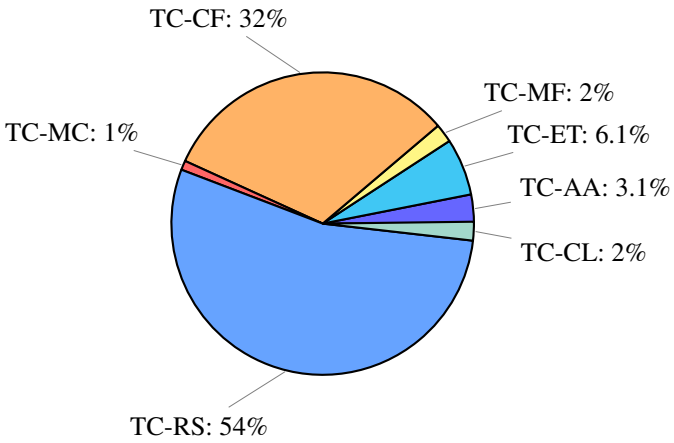

Figura 9. Distribuição dos mecanismos de controle adaptativo

para o ES. A técnica de *cluster* foi aplicado nos algoritmos de ECO e PSO. Nesta revisão não foram encontrados trabalhos envolvendo o controle adaptativo dos parâmetros do algoritmo BA.

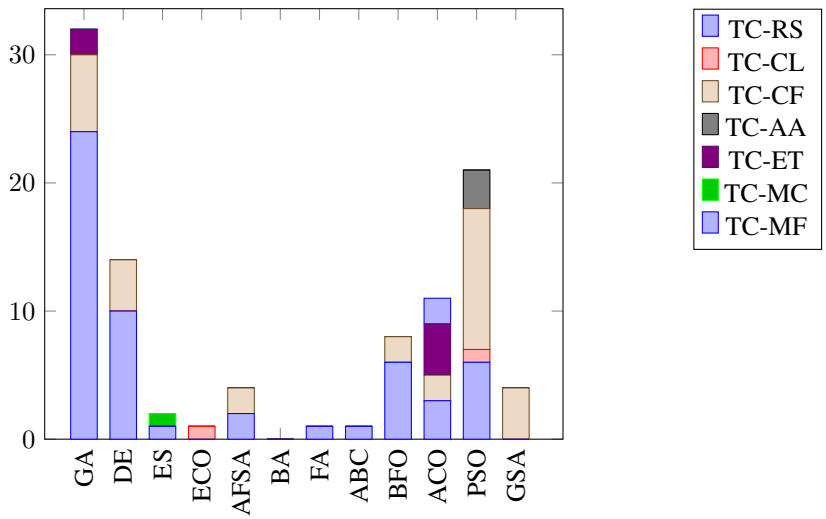

Figura 10. Distribuição das técnicas de controle adaptativo por algoritmo

7 Conclusão

Neste artigo foi apresentada uma revisão considerando o controle *online* de parâmetros em algoritmos bio-inspirados. No primeiro momento é apresentada uma revisão das taxonomias de controle de parâmetros encontradas na literatura, sendo em seguida definida uma taxonomia estendida baseada nos trabalhos de Eiben (1999) e Zhang (2012). A taxonomia proposta tem como objetivo destacar a técnica que está sendo utilizada no controle de parâmetros. Também são apresentados os principais algoritmos bio-inspirados utilizados na revisão.

O controle eficiente de parâmetros tem uma grande importância devido aos parâmetros definirem o comportamento, guiarem a busca e, consequentemente, interferirem na qualidade das soluções encontradas. As técnicas utilizadas para o controle de parâmetros são diversas. A presente revisão teve como objetivo identificar e classificar as técnicas encontradas na literatura. A classificação foi realizada em três categorias distintas: o controle determinista, o agregado e o adaptativo. Pela quantidade de diferentes técnicas de controle adaptativo, esta categoria precisou ser subdividida. As técnicas de controle determinista e controle adaptativo são as mais utilizadas dentre as técnicas revisadas. Porém as técnicas de controle adaptativo de parâmetros se destacam pelo fato de utilizarem informações de *feedback* da busca para guiar na escolha dos valores ótimos. Dentre as diferentes técnicas de controle adaptativo, a técnica mais encontrada na revisão da literatura é a técnica de regras simples. Outra técnica que também se destacou na revisão foi o controle *fuzzy*, demonstrando ser promissora no controle de parâmetros. Uma técnica muito pouco explorada e que apresentou uma boa motivação são os autômatos de aprendizagem. O autômato utiliza um processo de aprendizagem para melhorar seu desempenho progressivamente com o passar do tempo. Como visto nesta revisão, existem diferentes maneiras de obter o controle *online* de parâmetros. Desta forma, esta revisão pode guiar futuros pesquisadores na escolha de uma técnica de controle de parâmetros.

Nesta revisão pôde-se observar que o controle de parâmetros não é aplicado para todos os parâmetros dos algoritmos mas apenas aos parâmetros mais importante do algoritmo. Os algoritmos de GA, PSO e DE são os que mais possuem trabalhos envolvendo o controle *online* de parâmetros.

Como trabalhos futuros pode-se apontar a realização de experimentos para avaliar a eficiência de cada uma das técnicas apresentadas, como também identificar a melhor técnica para um determinado algoritmo e as suas respectivas limitações. Outra sugestão é o desenvolvimento de novas estratégias de controle, sendo diferentes das técnicas apresentadas nesta revisão, ou até mesmo, aplicar uma combinação entre elas.

Agradecimentos

Os autores gostariam de agradecer à Fundação de Amparo a Pesquisa e Inovação do Estado de Santa Catarina (FAPESC) pelo apoio financeiro, como também a Universidade do Estado de Santa Catarina (UDESC).

Referências

[1] A. A. Abshouri, M. R. Meybodi, and A. Bakhtiary. New firefly algorithm based on multi swarm & learning automata in dynamic environments. In *IEEE proceedings*, volume 13, pages 989–993, 2011.

[2] A. Aleti. Designing automotive embedded systems with adaptive genetic algorithms. *Automated Software Engineering*, pages 1–42, 2014.

[3] A. Aleti and I. Moser. Predictive parameter control. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 561–568. ACM, 2011.

[4] A. Aleti and I. Moser. Entropy-based adaptive range parameter control for evolutionary algorithms. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, GECCO '13, pages 1501–1508, 2013.

[5] A. Aleti and I. Moser. Studying feedback mechanisms for adaptive parameter control in evolutionary algorithms. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 3117–3124, 2013.

[6] A. Aleti, I. Moser, and S. Mostaghim. Adaptive range parameter control. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[7] A. Alfi. Particle swarm optimization algorithm with dynamic inertia weight for online parameter identification applied to lorenz chaotic system. *Int J Innov Comput Inf Control*, 8:1191 – 1204, 2012.

[8] M. M. Ali and A. Törn. Population set-based global optimization algorithms: Some modifications and numerical studies. *Comput. Oper. Res.*, 31(10):1703–1725, Sept. 2004.

[9] P. J. Angeline. Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.

[10] H. Askari and S.-H. Zahiri. Decision function estimation using intelligent gravitational search algorithm. *International Journal of Machine Learning and Cybernetics*, 3(2):163–172, 2012.

[11] D. Aydin, S. Özyön, C. Yaşar, and T. Liao. Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem. *International Journal of Electrical Power & Energy Systems*, 54(0):144–153, 2014.

[12] T. Back and M. Schutz. Intelligent mutation rate control in canonical genetic algorithms. In *Foundation of Intelligent Systems 9th International Symposium, ISMIS '96*, pages 158–167. Springer, 1996.

[13] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, mar 1993.

[14] P. Berkhin. A survey of clustering data mining techniques. In J. Kogan, C. Nicholas, and M. Teboulle, editors, *Grouping Multidimensional Data*, pages 25–71. Springer Berlin Heidelberg, 2006.

[15] H.-G. Beyer. Toward a theory of evolution strategies: Self-adaptation, 1995.

[16] H.-G. Beyer and H.-P. Schwefel. Evolution strategies–a comprehensive introduction. *Natural computing*, 1(1):3–52, 2002.

[17] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM COMPUTING SURVEYS*, pages 268–308, 2003.

[18] D. Boeringer and D. Werner. Adaptive mutation parameter toggling genetic algorithm for phase-only array synthesis. *Electronics Letters*, 38(25):1618–1619, 2002.

[19] K. Boudjelaba, F. Ros, and D. Chikouche. An efficient hybrid genetic algorithm to design finite impulse response filters. *Expert Systems with Applications*, 41(13):5917 – 5937, 2014.

[20] I. Boussaid, J. Lepagnot, and P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82 – 117, 2013.

[21] J. Brest, V. Zumer, and M. Maucec. Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 215–222, 2006.

[22] Z. Cai and H. Huang. Ant colony optimization algorithm based on adaptive weight and volatility parameters. In *Intelligent Information Technology Application, 2008. IITA '08. Second International Symposium on*, volume 2, pages 75–79, Dec 2008.

[23] O. Castillo, H. Neyoy, J. Soria, M. García, and F. Valdez. Dynamic fuzzy logic parameter tuning for aco and its application in the fuzzy logic control of an autonomous mobile robot. *International Journal of Advanced Robotic Systems*, 10, 2013.

[24] A. Chatterjee and P. Siarry. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers and Operations Research*, 33(3):859 – 871, 2006.

[25] B. Chen, B. Yan, and Y. Wang. Research on multiple kill vehicles firepower distribution strategy based on adjust genetic algorithm. In *Control and Decision Conference (CCDC), 2013 25th Chinese*, pages 3582–3586, May 2013.

[26] H. Chen, Y. Zhu, and K. Hu. Adaptive bacterial foraging optimization. In *Abstract and Applied Analysis*, volume 2011. Hindawi Publishing Corporation, 2011.

[27] L. Chen, S. Zhao, W. Zhu, Y. Liu, and W. Zhang. A self-adaptive differential evolution algorithm for parameters identification of stochastic genetic regulatory networks with random delays. *Arabian Journal for Science and Engineering*, 39(2):821–835, 2014.

[28] Z. CHEN, Y. ZHOU, and M. LU. A simplified adaptive bat algorithm based on frequency. *Journal of Computational Information Systems*, 9(16):6451– 6458, 2013.

[29] S. Chusanapiputt, D. Nualhong, S. Jantarang, and S. Phoomvuthisarn. Selective self-adaptive approach to ant system for solving unit commitment problem. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1729–1736. ACM, 2006.

[30] D. Dasgupta, S. Yu, and F. Nino. Recent advances in artificial immune systems: models and applications. *Applied Soft Computing*, 11(2):1574–1587, 2011.

[31] S. Dasgupta, S. Das, A. Abraham, and A. Biswas. Adaptive computational chemotaxis in bacterial foraging optimization: An analysis. *Evolutionary Computation, IEEE Transactions on*, 13(4):919–941, Aug 2009.

[32] T. Datta, I. S. Misra, B. B. Mangaraj, and S. Imtiaj. Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence. *Progress In Electromagnetics Research C*, 1:143–157, 2008.

[33] L. Davis et al. *Handbook of genetic algorithms*, volume 115. Van Nostrand Reinhold New York, 1991.

[34] K. De Jong. *Evolutionary Computation: A Unified Approach*. Bradford Book. Mit Press, 2006.

[35] K. De Jong. Parameter setting in eas: a 30 year perspective. pages 1–18. Springer Berlin Heidelberg, 2007.

[36] Y.-Y. Ding and X.-Y. Wang. Real-coded adaptive genetic algorithm applied to pid parameter optimization on a 6r manipulators. In *Natural Computation, 2008. ICNC 08. Fourth International Conference on*, volume 1, pages 635–639, 2008.

[37] F. Dobslaw. A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks. In *Proceeding of the International Conference on Computer Mathematics and Natural Computing 2010*. WASET, 2010.

[38] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2. IEEE, 1999.

[39] E.-N. Dragoi, S. Curteanu, A.-I. Galaction, and D. Cascaval. Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process. *Applied Soft Computing*, 13(1):222 – 238, 2013.

[40] R. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 94–100 vol. 1, 2001.

[41] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on*, 3(2):124–141, 1999.

[42] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 2011.

[43] O. K. Erol and I. Eksin. A new optimization method: Big bang-big crunch. *Advances in Engineering Software*, 37(2):106 – 111, 2006.

[44] I. Farhat and M. El-Hawary. Dynamic adaptive bacterial foraging algorithm for optimum economic dispatch with valve-point effects and wind power. *Generation, Transmission Distribution, IET*, 4(9):989–999, September 2010.

[45] J. A. Fernandez-Prieto, J. Canada-Bago, M. A. Gadeo-Martos, and J. R. Velasco. Optimisation of control parameters for genetic algorithms to test computer networks under realistic traffic loads. *Appl. Soft Comput.*, 11(4):3744–3752, June 2011.

[46] I. Fister, X.-S. Yang, J. Brest, and I. J. Fister. 4 - memetic self-adaptive firefly algorithm. In *Swarm Intelligence and Bio-Inspired Computation*, pages 73 – 102. Elsevier, Oxford, 2013.

[47] I. Fister, Jr., X.-S. Yang, I. Fister, J. Brest, and D. Fister. A Brief Review of Nature-Inspired Algorithms for Optimization. *ArXiv e-prints*, July 2013.

[48] Z. W. Geem, J. H. Kim, and G. Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[49] H. Hamza, A. Kamel, and K. Shams. Software effort estimation using artificial neural networks: A survey of the current practices. In *Information Technology: New Generations (ITNG), 2013 Tenth International Conference on*, pages 731–733, April 2013.

[50] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001.

[51] A. Hashemi and M. Meybodi. A note on the learning automata based algorithms for adaptive parameter selection in pso. *Applied Soft Computing*, 11(1):689 – 705, 2011.

[52] A. Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222(0):175 – 184, 2013.

[53] S. He et al. Fuzzy clustering with improved artificial fish swarm algorithm. In *2009 International Joint Conference on Computational Sciences and Optimization*, volume 2, pages 317–321, 2009.

[54] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in evolutionary computation: A survey. In *In Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC 97*, pages 65–69. IEEE Press, 1997.

[55] X. Hongrui, L. Ran, G. Jianli, and W. Hongru. An adaptive meta-cognitive artificial fish school algorithm. In *Proceedings of the 2009 International Forum on Information Technology and Applications - Volume 01*, IFITA '09, pages 594–597. IEEE Computer Society, 2009.

[56] Z. Huang. and Y. Chen. An improved differential evolution algorithm based on adaptive parameter. *Journal of Control Science and Engineering*, 2013, 2013.

[57] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[58] J. Jantzen. *Fuzzy Control*, pages 47–70. John Wiley e Sons, Ltd, 2007.

[59] B. Jiao, Z. Lian, and X. Gu. A dynamic inertia weight particle swarm optimization algorithm. *Chaos, Solitons and Fractals*, 37(3):698 – 705, 2008.

[60] L. Jiye, C. Xihong, L. Qiang, and S. Jizhe. Prediction of satellite clock errors using ls-svm optimized by improved artificial fish swarm algorithm. In *Signal Processing, Communication and Computing (ICSPCC), 2013 IEEE International Conference on*, pages 1–5, Aug 2013.

[61] N. R. Kamrath, B. W. Goldman, and D. R. Tauritz. Using supportive coevolution to evolve self-configuring crossover. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion*, GECCO '13 Companion. ACM, 2013.

[62] D. Karaboga. An idea based on honey bee swarm for numerical optimization. *Techn. Rep. TR06, Erciyes Univ. Press, Erciyes*, 2005.

[63] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation*, 214(1):108 – 132, 2009.

[64] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[65] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Evolutionary Computation Series. Morgan Kaufmann Publishers, 2001.

[66] M. Khichane, P. Albert, and C. Solnon. Learning and intelligent optimization. chapter An ACO-Based Reactive Framework for Ant Colony Optimization: First Experiments on Constraint Satisfaction Problems, pages 119–133. Springer-Verlag, Berlin, Heidelberg, 2009.

[67] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[68] M. Kotinis. Improving a multi-objective differential evolution optimizer using fuzzy adaptation and k-medoids clustering. *Soft Computing*, 18(4):757–771, 2014.

[69] D. Kovačević, N. Mladenović, B. Petrović, and P. Milošević. De-vns: Self-adaptive differential evolution with crossover neighborhood search for continuous global optimization. *Computers & Operations Research*, (0):–, 2014.

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

[71] O. Kramer. Evolutionary self-adaptation: a survey of operators and strategy parameters. *Evolutionary Intelligence*, 3(2):51–65, 2010.

[72] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. Lopes. A survey of swarm algorithms applied to discrete optimization problems. In *Swarm Intelligence and Bio-Inspired Computation*, pages 169 – 191. Elsevier, Oxford, 2013.

[73] J. V. Kumar, D. V. Kumar, and K. Edukondalu. Strategic bidding using fuzzy adaptive gravitational search algorithm in a pool based electricity market. *Applied Soft Computing*, 13(5):2445 – 2455, 2013.

[74] S. W. Leung, X. Zhang, and S.-Y. Yuen. Multiobjective differential evolution algorithm with opposition-based parameter control. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, 2012.

[75] C. Li and S. Yang. An adaptive learning particle swarm optimizer for function optimization. In *Evolutionary Computation, 2009. CEC09. IEEE Congress on*, pages 381–388. IEEE, 2009.

[76] H. Li and P. Li. Self-adaptive ant colony optimization for construction time-cost optimization. *Kybernetes*, 42(8):1181–1194, 2013.

[77] R. Li and X. Chang. A modified genetic algorithm with multiple subpopulations and dynamic parameters applied in cvar model. In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 151–151, 2006.

[78] Y. Li. A novel aco with average entropy. *Journal of Software Engineering & Applications*, 2(5), 2009.

[79] X. Liang, W. Li, Y. Zhang, and M. Zhou. An adaptive particle swarm optimization method based on clustering. *Soft Computing*, pages 1–18, 2014.

[80] M. M. Linda and N. K. Nair. A new-fangled adaptive mutation breeder genetic optimization of global multi-machine power system stabilizer. *International Journal of Electrical Power and Energy Systems*, 44(1):249 – 258, 2013.

[81] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6):448–462, 2005.

[82] R. Majhi, G. Panda, B. Majhi, and G. Sahoo. Efficient prediction of stock market indices using adaptive bacterial foraging optimization (abfo) and bfo based techniques. *Expert Systems with Applications*, 36(6):10097 – 10104, 2009.

[83] R. Mallipeddi and P. Suganthan. Differential evolution algorithm with ensemble of parameters and mutation and crossover strategies. In *Swarm, Evolutionary, and Memetic Computing*, volume 6466 of *Lecture Notes in Computer Science*, pages 71–78. Springer Berlin Heidelberg, 2010.

[84] V. Marques and F. Gomide. Parameter control of metaheuristics with genetic fuzzy systems. *Evolutionary Intelligence*, pages 183–202, 2011.

[85] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez. Optimal design of fuzzy classification systems using {PSO} with dynamic parameter adaptation through fuzzy logic. *Expert Systems with Applications*, 40(8):3196 – 3206, 2013.

[86] D. Merkle, M. Middendorf, and H. Schmeck. Ant colony optimization for resource-constrained project scheduling. *Evolutionary Computation, IEEE Transactions on*, 6(4):333–346, Aug 2002.

[87] B. Meyer. Convergence control in ACO. In *Genetic and Evolutionary Computation Conference (GECCO), Seattle, WA, late-breaking paper available on CD*, 2004.

[88] E. Mezura Montes and E. A. López-Dávila. Adaptation and local search in the modified bacterial foraging algorithm for constrained optimization. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8. IEEE, 2012.

[89] S. Mishra. A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. *Evolutionary Computation, IEEE Transactions on*, 9(1):61–73, Feb 2005.

[90] I. Montalvo, J. Izquierdo, R. Pérez-García, and M. Herrera. Improved performance of pso with self-adaptive parameters for computing the optimal design of water supply systems. *Engineering Applications of Artificial Intelligence*, 23(5):727 – 735, 2010.

[91] V. Nannen, S. Smit, and A. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. In *Parallel Problem Solving from Nature – PPSN X*, volume 5199 of *Lecture Notes in Computer Science*, pages 528–538. Springer Berlin Heidelberg, 2008.

[92] M. Neshat. Faipso: fuzzy adaptive informed particle swarm optimization. *Neural Computing and Applications*, 23(1):95–116, 2013.

[93] M. Neshat, G. Sepidnam, M. Sargolzaei, and A. Toosi. Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, pages 1–33, 2012.

[94] T. Niknam and E. Azad Farsani. A hybrid self-adaptive particle swarm optimization and modified shuffled frog leaping algorithm for distribution feeder reconfiguration. *Eng. Appl. Artif. Intell.*, 23(8):1340–1349, dec 2010.

[95] T. Niknam, E. Azadfarsani, and M. Jabbari. A new hybrid evolutionary algorithm based on new fuzzy adaptive {PSO} and {NM} algorithms for distribution feeder reconfiguration. *Energy Conversion and Management*, 54(1):7 – 16, 2012.

[96] T. Niknam, R. Azizipanah-Abarghooee, and A. Roosta. Reserve constrained dynamic economic dispatch: a new fast self-adaptive modified firefly algorithm. *Systems Journal, IEEE*, 6(4):635–646, 2012.

[97] T. Niknam, M. Narimani, R. Azizipanah-Abarghooee, and B. Bahmani-Firouzi. Multiobjective optimal reactive power dispatch and voltage control: A new opposition-based self-adaptive modified gravitational search algorithm. *Systems Journal, IEEE*, 7(4):742–753, Dec 2013.

[98] M. Obaidat, G. Papadimitriou, and A. Pomportsis. Guest editorial learning automata: theory, paradigms, and applications. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 32(6):706–709, Dec 2002.

[99] A. Okafor. *Entropy based techniques with applications in data mining*. PhD thesis, University of Florida, 2005.

[100] F. Olivas, F. Valdez, and O. Castillo. Particle swarm optimization with dynamic parameter adaptation using interval type-2 fuzzy logic for benchmark mathematical functions. In *Nature and Biologically Inspired Computing (NaBIC), 2013 World Congress on*, pages 36–40, Aug 2013.

[101] E. Osaba, E. Onieva, R. Carballedo, F. Diaz, A. Perallos, and X. Zhang. A multi-crossover and adaptive island based population algorithm for solving routing problems. *Journal of Zhejiang University SCIENCE C*, 14(11):815–821, 2013.

[102] Z. Pan and L. Chen. Minimization of binary decision diagrams by adaptive hierarchy genetic algorithm and its application in circuit test generation. In *Advances in Mechanical and Electronic Engineering*, volume 178, pages 175–180. Springer Berlin Heidelberg, 2013.

[103] B. Panigrahi and V. R. Pandi. Congestion management using adaptive bacterial foraging algorithm. *Energy Conversion and Management*, 50(5):1202 – 1209, 2009.

[104] R. Parpinelli and H. Lopes. A hierarchical clustering strategy to improve the biological plausibility of an ecology-based evolutionary algorithm. In *Advances in Artificial Intelligence – IBERAMIA 2012*, volume 7637 of *Lecture Notes in Computer Science*, pages 310–319. Springer Berlin Heidelberg, 2012.

[105] R. S. Parpinelli and H. S. Lopes. An eco-inspired evolutionary algorithm applied to numerical optimization. In *NaBIC*, pages 466–471. IEEE, 2011.

[106] R. S. Parpinelli and H. S. Lopes. New inspirations in swarm intelligence; a survey. *Int. J. Bio-Inspired Comput.*, 3(1):1–16, Feb. 2011.

[107] K. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE*, 22(3):52–67, Jun 2002.

[108] Y. Peng, X. Luo, and W. Wei. A new fuzzy adaptive simulated annealing genetic algorithm and its convergence analysis and convergence rate estimation. *International Journal of Control, Automation and Systems*, 12(3):670–679, 2014.

[109] D. Pham and A. Ghanbarzadeh. Multi-objective optimisation using the bees algorithm. In *Proceedings of IPROMS 2007 Conference*, 2007.

[110] A. P. Piotrowski. Adaptive memetic differential evolution with global and local neighborhood-based mutation operators. *Information Sciences*, 241(0):164 – 194, 2013.

[111] J. L. Ponz-Tienda, V. Yepes, E. Pellicer, and J. Moreno-Flores. The resource leveling problem with multiple resources using an adaptive genetic algorithm. *Automation in Construction*, 29(0):161 – 172, 2013.

[112] R. Precup, R. David, E. Petriu, S. Preitl, and M. Radac. Novel adaptive gravitational search algorithm for fuzzy controlled servo systems. *Industrial Informatics, IEEE Transactions on*, 8(4):791–800, Nov 2012.

[113] R.-E. Precup, R.-C. David, E. Petriu, S. Preitl, and M.-B. Radac. Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems. *Control Theory Applications, IET*, 7(1):99–107, Jan 2013.

[114] B. Rajakumar and A. George. Apoga: An adaptive population pool size based genetic algorithm. *{AASRI} Procedia*, 4(0):288 – 296, 2013.

[115] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. Gsa: A gravitational search algorithm. *Information Sciences*, 179(13):2232 – 2248, 2009. Special Section on High Order Fuzzy Sets.

[116] V. Rashtchi, A. Bayat, and H. Vahedi. Adaptive step length bacterial foraging algorithm. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 322–326, Nov 2009.

[117] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973.

[118] A. Rezaee Jordehi and J. Jasni. Parameter selection in particle swarm optimisation: a survey. *Journal of Experimental e Theoretical Artificial Intelligence*, 2013.

[119] E. Ridge. Design of experiments for the tuning of optimisation algorithms. Technical report, The University of York, 2007.

[120] L. Rokach. A survey of clustering algorithms. In O. Maimon and L. Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 269–298. Springer US, 2010.

[121] A. Roy, P. Rakshit, A. Konar, S. Bhattacharya, E. Kim, and A. Nagar. Adaptive firefly algorithm for nonholonomic motion planning of car-like system. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2162–2169, June 2013.

[122] F.-S. Saeidi-Khabisi and E. Rashedi. Fuzzy gravitational search algorithm. In *Computer and Knowledge Engineering (ICCKE), 2012 2nd International eConference on*, pages 156–160, 2012.

[123] P. Sathya and R. Kayalvizhi. Optimal segmentation of brain mri based on adaptive bacterial foraging algorithm. *Neurocomputing*, 74(14–15):2299 – 2313, 2011.

[124] R. Selvarasu and C. Rajan. Self adaptive firefly algorithm based transmission loss minimization using tcsc. In *Advanced Nanomaterials and Emerging Engineering Technologies (ICANMEET), 2013 International Conference on*, pages 683–688, July 2013.

[125] C. E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[126] Y. Shi and R. Eberhart. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages –1950 Vol. 3, 1999.

[127] C. Simons and I. Parmee. Dynamic parameter control of interactive local search in uml software design. In *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pages 3397–3404, 2010.

[128] M. Smetek and B. Trawinski. Investigation of self-adapting genetic algorithms using some multimodal benchmark functions. In *Computational Collective Intelligence. Technologies and Applications*, volume 6922 of *Lecture Notes in Computer Science*, pages 213–223. Springer Berlin Heidelberg, 2011.

[129] S. K. Smit and A. E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *In Proceedings of the IEEE Congress on Evolutionary Computation (CEC*, pages 399–406, 2009.

[130] J. E. Smith and T. C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, pages 81–87, 1997.

[131] A. Sombra, F. Valdez, P. Melin, and O. Castillo. A new gravitational search algorithm using fuzzy logic to parameter adaptation. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1068–1074, 2013.

[132] M. Srinivas and L. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 24(4):656–667, 1994.

[133] R. Storn and K. Price. Differential evolution : A simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, Dec. 1997.

[134] R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 71–78, June 2013.

[135] M. Tarokh and X. Zhang. Real-time motion tracking of robot manipulators using adaptive genetic algorithms. *Journal of Intelligent & Robotic Systems*, 74(3-4):697–708, 2014.

[136] J. Teo. Exploring dynamic self-adaptive populations in differential evolution. *Soft Computing*, 10(8):673–686, 2006.

[137] W. Tian and Y. Tian. An improved artificial fish swarm algorithm for resource leveling. In *2009 International Conference on Management and Service Science*, pages 1–4, 2009.

[138] A. Tuson and P. Ross. Adapting operator settings in genetic algorithms. *Evol. Comput.*, pages 161–184, June 1998.

[139] F. Vafaee and P. Nelson. A genetic algorithm that incorporates an adaptive mutation based on an evolutionary model. In *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, pages 101–107, 2009.

[140] P. Vajda, A. E. Eiben, and W. Hordijk. Parameter control methods for selection operators in genetic algorithms. In *Parallel Problem Solving from Nature–PPSN X*, pages 620–630. Springer, 2008.

[141] C. Venkaiah and D. V. Kumar. Fuzzy adaptive bacterial foraging congestion management using sensitivity based optimal active power re-scheduling of generators. *Applied Soft Computing*, 11(8):4921 – 4930, 2011.

[142] J. WANG. Particle swarm optimization with adaptive parameter control and opposition. *Journal of Computational Information Systems*, 7:4463– 4470, 2011.

[143] X. Wang and S. Zhao. Differential evolution algorithm with self-adaptive population resizing mechanism. *Mathematical Problems in Engineering*, 2013, 2013.

[144] J. Q. X. Li, Z. Shao. An optimizing method base on autonomous animates: fish- swarm algorithm. pages 32–38, 2002.

[145] X. Xu and H.-l. Chen. Adaptive computational chemotaxis based on field in bacterial foraging optimization. *Soft Computing*, 18(4):797–807, 2014.

[146] X. Yan, Y. Zhu, J. Wu, and H. Chen. An improved firefly algorithm with adaptive strategies. *Advanced Science Letters*, 16(1):249–254, 2012.

[147] X. Yan, Y. Zhu, H. Zhang, H. Chen, and B. Niu. An adaptive bacterial foraging optimization algorithm with lifecycle and social learning. *Discrete Dynamics in Nature and Society*, 2012, 2012.

[148] K. Yang, J. Zheng, M. Yang, R. Zhou, and G. Liu. Adaptive genetic algorithm for daily optimal operation of cascade reservoirs and its improvement strategy. *Water Resources Management*, 27(12):4209–4235, 2013.

[149] L. Yang, D. Widyantoro, T. Ioerger, and J. Yen. An entropy-based adaptive genetic algorithm for learning classification rules. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 2, pages 790–796 vol. 2, 2001.

[150] X. Yang, J. Yuan, J. Yuan, and H. Mao. A modified particle swarm optimizer with dynamic adaptation. *Applied Mathematics and Computation*, 189(2):1205 – 1213, 2007.

[151] X.-S. Yang. *Nature-inspired metaheuristic algorithms*. Luniver Press, 2010.

[152] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, volume 284 of *Studies in Computational Intelligence*, pages 65–74. Springer Berlin Heidelberg, 2010.

[153] K. Yasuda, K. Yazawa, and M. Motoki. Particle swarm optimization with parameter self-adjusting mechanism. *IEEJ Transactions on Electrical and Electronic Engineering*, 5(2):256–257, 2010.

[154] D. Yazdani, H. Nabizadeh, E. M. Kosari, and A. N. Toosi. Color quantization using modified artificial fish swarm algorithm. In *AI 2011: Advances in Artificial Intelligence*, pages 382–391. Springer, 2011.

[155] D. Yazdani, A. Nadjaran Toosi, and M. Meybodi. Fuzzy adaptive artificial fish swarm algorithm. In *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science*, pages 334–343. Springer Berlin Heidelberg, 2011.

[156] S. Yılmaz, E. Ugur Kucuksille, and Y. Cengiz. Modified bat algorithm. *Electronics & Electrical Engineering*, 20(2), 2014.

[157] P. Yong. An improved artificial fish swarm algorithm for optimal operation of cascade reservoirs. *Journal of computers*, 6(4), 2011.

[158] S. Zahiri. Fuzzy gravitational search algorithm an approach for data mining. *Iranian Journal of Fuzzy Systems*, 2012.

[159] H. Zhang, J. Zhou, N. Fang, R. Zhang, and Y. Zhang. An efficient multi-objective adaptive differential evolution with chaotic neuron network and its application on long-term hydropower operation with considering ecological environment problem. *International Journal of Electrical Power & Energy Systems*, 45(1):60 – 70, 2013.

[160] J. Zhang, W.-N. Chen, Z.-H. Zhan, W.-J. Yu, Y.-L. Li, N. Chen, and Q. Zhou. A survey on algorithm adaptation in evolutionary computation. *Frontiers of Electrical and Electronic Engineering*, 7(1):16–31, 2012.

[161] Y. Zhong, L. Zhao, and L. Zhang. An adaptive differential evolution endmember extraction algorithm for hyperspectral remote sensing imagery. *Geoscience and Remote Sensing Letters, IEEE*, 11(6):1061–1065, June 2014.

[162] K. Zielinski and R. Laur. Adaptive parameter setting for a multi-objective particle swarm optimization algorithm. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3019–3026, Sept 2007.