Um Estudo Sobre a Configuração Automática do Algoritmo de Evolução Diferencial

Rodrigo César Pedrosa Silva

Dissertação submetida à
Escola de Engenharia
Universidade Federal de Minas Gerais
para obtenção do título de Mestre em Engenharia Elétrica

Dedico este trabalho a minha irmã, Ludmila, que sempre ri das minhas piadas.

Um Estudo Sobre a Configuração Automática do Algoritmo de Evolução Diferencial

Resumo

O grande desenvolvimento na área de algoritmos evolutivos nas últimas décadas tem aumentado o domínio de aplicações dessas ferramentas e melhorado seu desempenho em diversas frentes. Em particular, o algoritmo de Evolução Diferencial (DE - Differential Evolution) tem se mostrado um otimizador simples e eficiente em diversos contextos. Apesar do sucesso, seu desempenho está diretamente relacionado à escolha adequada dos operadores de variação e dos parâmetros que controlam o funcionamento destes operadores. Para aumentar a robustez do método e facilitar a sua utilização para o usuário comum, tem-se buscado cada vez mais por métodos de configuração automática. Existem na literatura diversos métodos para configuração de operadores e parâmetros. A fim de entender os efeitos dessas abordagens no desempenho do DE, neste trabalho é apresentada uma análise experimental criteriosa das principais abordagens existentes. Além disso é apresentada uma nova abordagem para a seleção de operadores baseada em aprendizado por reforço. Os resultados mostram que abordagens bem simples são capazes de trazer melhoras significativas ao desempenho do DE.

A Study on Self-configuration in the Differential Evolution Algorithm

Abstract

The great development in the area of evolutionary algorithms in recent decades has increased the range of applications of these tools and improved its performance in different fronts. In particular, the Differential Evolution (DE) algorithm has proven to be a simple and efficient optimizer in several contexts. Despite its success, its performance is closely related to the choice of variation operators and the parameters which control these operators. To increase the robustness of the method and the ease of use for the average user, the pursuit for methods of self-configuration has been increasing as well. There are several methods in the literature for setting parameters and operators. In order to understand the effects of these approaches on the performance of DE, this paper presents a thorough experimental analysis of the main existing approaches. Furthermore, a new approach is presented for selecting operators based on reinforcement learning. The results show that simple approaches are able to bring significant improvements to the performance of DE.

Acknowledgements

Agradeço ao meu orientador, Frederico Gadelha Guimarães, pela orientação científica criteriosa e crítica, estimulando e dando o tempo para uma construção pessoal deste trabalho. A disponibilidade que sempre manifestou e a empatia com que recebeu as minhas ideias foram os estímulos que me permitiram vencer as dificuldades de mais esta etapa.

Agradeço aos amigos e grandes colaboradores, Alan de Freitas, Ricardo Prado e Rodolfo Lopes, pelo companheirismo tanto na academia quanto fora e pelas discussões geradas em nosso grupo de pesquisa informal sobre POPES (*Parallel Optimization and Evolution Strategies*). À Camila Aparecida pela força com as vírgulas e pela injeção de ânimo na conclusão deste trabalho. E ao professor Felipe Campelo, pelo reforço nos ensinamentos em boas práticas científicas e pela valiosa e fundamental ajuda no planejamento experimental deste trabalho.

Agradeço aos meus pais, Marcelo e Maria Lúcia, através dos quais tive meus primeiros contatos com a ciência, que me ensinaram e me permitiram seguir o caminho que escolhesse, sempre tornando a minha caminhada o mais suave possível.

Ao CNPq e à CAPES pelo apoio financeiro, sem o qual esse trabalho não teria sido possível.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A crescente demanda pela diminuição dos custos e aumento de desempenho nas mais variadas aplicações em computação e engenharia tem levado a comunidade científica a olhar com atenção para métodos rigorosos de tomada de decisão. A tarefa de tomada de decisão implica na escolha da "melhor" dentre várias ou até infinitas alternativas. Para guiar tal decisão cria-se uma descrição, ou modelo, que permite dizer o quão boa uma alternativa é em relação a um determinado objetivo. Os métodos de otimização lidam então com a seleção da melhor alternativa dado um ou vários objetivos.

Como ilustrado na Figura 1.1, em um problema de otimização possuímos um modelo, que nos permite avaliar uma possível alternativa, ou solução candidata, e a descrição do que se deseja na saída. Assim, a tarefa é encontrar a alternativa que nos leva à saída com as características previamente determinadas. Um exemplo é o problema da diversidade máxima, no qual devemos encontrar o subconjunto mais representativo, ou seja, de maior diversidade. Nele temos uma fórmula (o modelo) para calcular a diversidade (a saída) de um determinado subconjunto. A propriedade desejada na saída é a otimalidade, isto é, a máxima diversidade. O objetivo é, então, encontrar a(s) entrada(s) que possua(m) essa propriedade.

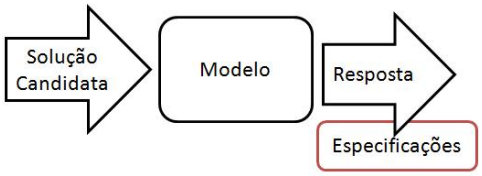


Figura 1.1: Problema de Otimização

Qual o menor percurso ligando um determinado conjunto de cidades? Que aplicações maximizam o lucro de uma empresa? Que polinômio melhor representa um determinado conjunto de dados? Estes são alguns problemas de tomada de decisão que podem ser modelados como um problema de otimização. Uma formulação genérica[1] para um problema de otimização pode ser dada por:

$$\mathbf{x}^* = \arg\min_x f(\mathbf{x}) \in \mathbb{R} \tag{1.1}$$

$$\text{sujeito a: } \begin{cases} \mathbf{x} \in \mathbb{R}^n \\ g_1(\mathbf{x}) \leq 0 \\ \vdots \\ g_m(\mathbf{x}) \leq 0 \end{cases} \tag{1.2}$$

Em que $\mathbf{x} \in \mathbb{R}^n$ é o vetor de variáveis de otimização, $f(.)$ a função objetivo que descreve o modelo, e as funções $g_i, i = \{1, ..., m\}$ as restrições que definem a região factível. Se não houver nenhuma restrição, o problema de otimização é chamado irrestrito.

Graças ao rápido desenvolvimento da computação no que diz respeito principalmente a desempenho, um dos principais temas da matemática e da ciência da computação vem sendo a criação de resolvedores automáticos de problemas (isto é, algoritmos). Nos últimos anos, estas técnicas adquiriram um bom nível de maturidade sendo utilizadas em uma grande variedade de indústrias, tais como, a indústria química, elétrica, de construção e aeroespacial. Contudo, com o avanço das tecnologias utilizadas em cada uma destas áreas assim como o aumento da própria demanda por produtos e processos mais eficientes, o tamanho e a complexidade dos problemas de otimização também tem aumentado. Dessa forma, a dificuldade em otimizar estes sistemas tem desafiado as capacidades das técnicas tradicionais e levado a comunidade científica a busca de novas abordagens.

Uma maneira razoável de se obter novas abordagens é buscar inspiração na Natureza. Uma de suas características mais surpreendentes é a enorme diversidade de seres vivos, adaptados a quase qualquer ambiente, desde o clima gélido dos polos às temperaturas

[1] A escolha pelo problema de minimização é arbitrária. Isso, não implica em perda de generalidade uma vez que $\max f(\mathbf{x}) = \min(-f(\mathbf{x}))$

escaldantes dos desertos. Isso se torna ainda mais extraordinário quando notamos que o ambiente muda continuamente e nem assim a vida desaparece. O processo que conduz o surgimento de estruturas orgânicas mais complexas, permitindo que os seres vivos se adaptem às mais variadas situações e ambientes, é conhecido como Evolução. E é exatamente este processo que vem inspirando uma grande família de métodos chamados de Algoritmos Evolutivos (AEs).

Os AEs são um tipo de método probabilístico que imitam o processo de evolução natural e, principalmente a partir dos anos 1980, têm sido utilizados para a resolução de problemas difíceis de otimização (Back, Hammel and Schwefel 1997). A base por trás de todos estes métodos é o mecanismo de seleção natural, que é o processo pelo qual indivíduos mais bem adaptados são selecionados, descrito em (Darwin 1859). Adaptando esta ideia para o processo de otimização, tem-se uma população formada por um conjunto de soluções candidatas do problema, estas soluções sofrem algum tipo de variação (essencialmente cruzamento e mutação) e têm sua adaptabilidade, ou *fitness*, medida através da função objetivo (o modelo). Soluções mais adaptadas, ou seja, soluções melhores, têm uma maior chance de sobreviver. À medida que esse processo, de variação e seleção, se repete, a população evolui gradativamente e soluções cada vez melhores são gestimadas. A figura 1.2 mostra o esquema de um algoritmo evolutivo básico.

A classe de algoritmos evolutivos apresenta várias famílias de métodos, dentre as quais, a família dos Algoritmos Genéticos (Goldberg 1989) é provavelmente a mais popular. Contudo, nos últimos anos, novas famílias de métodos têm sido desenvolvidas, tais como algoritmos de Sistemas Imunes Artificiais (AIS - Artificial Immune Systems)(Castro 2002), algoritmos de Otimização por Enxame de Partículas (PSO - Particle Swarm Optimization)(Kennedy and Eberhart 1995) e algoritmos de Evolução Diferencial (DE - Differential Evolution) (Storn and Price 1995). As principais vantagens do uso de AEs são sua fácil adaptação a uma grande variedade de problemas, fácil implementação e aplicação, combinadas com desempenho robusto (apesar desse fator depender da classe do problema) e com suas características de busca global (Back, Hammel and Schwefel 1997).

Nas últimas décadas, os avanços obtidos na área dos AEs aumentaram o domínio de aplicações dessas ferramentas e melhoraram seu desempenho em diversos contextos. Em particular, o algoritmo de Evolução Diferencial (Storn and Price 1995), proposto em meados da década de 1990, vem se destacando, apresentando resultados notáveis, como por exemplo, nas competições da IEEE International Conference on Evolutionary Computation (CEC). O DE obteve primeiro lugar na competição de otimização de parâmetros

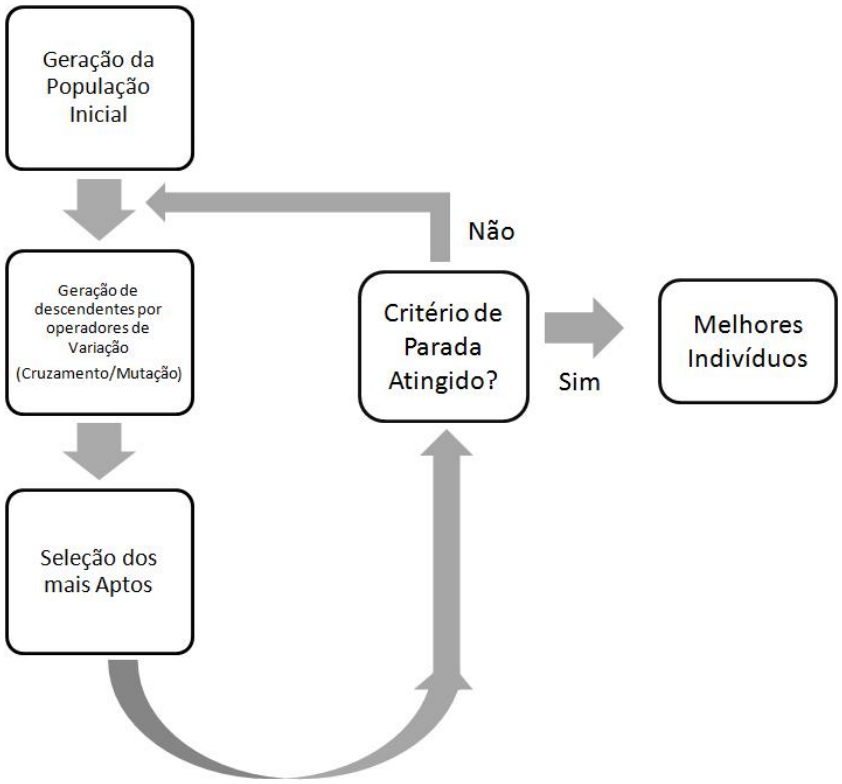


Figura 1.2: Algoritmo Evolutivo Básico

reais com restrições do CEC 2006, terceiro lugar na competição de otimização global em larga escala do CEC 2007 e primeiro lugar na competição de algoritmos evolutivos em ambientes dinâmicos e incertos do CEC 2009. Graças a seu alto desempenho e facilidade de implementação, o DE se tornou bastante popular quase imediatamente após sua definição. Atualmente tem sido usado com sucesso numa grande variedade de aplicações, como agrupamento de dados (Sai, Vinaya, Govardhan and Satapathy 2010), sequenciamento de tarefas (Onwubolu and Davendra 2006), projeto de filtros digitais (Karaboga 2005) e otimização de sistemas de reservatório (Reddy and Kumar 2007).

O DE, assim como os outros algoritmos evolutivos, já demonstrou sua habilidade em tratar problemas de otimização fora do alcance de métodos tradicionais (DaCosta, Fialho, Schoenauer and Sebag 2008). Contudo, o desempenho destes métodos na resolução de um determinado problema está diretamente ligado à correta definição de alguns parâmetros (como o tamanho da população, o tipo dos operadores de variação e as respectivas taxas de aplicação, tipos dos mecanismos de seleção, dentre outros). O trabalho de configurar corretamente parâmetros e operadores, além de ser uma tarefa normalmente demorada, em muitos casos exige que os usuários já possuam alguma

experiência em AEs.

Para tornar os AEs mais flexíveis, no sentido de mantê-los robustos a um maior número de problemas sem a necessidade de intervenção do usuário, uma área de pesquisa que tem crescido bastante na área de computação evolutiva é o de métodos autoconfiguráveis (Maturana and Saubion 2008, Thierens 2005, Fialho, Costa, Schoenauer and Sebag 2009, Qin, Huang and Suganthan 2009). Nestes métodos, o próprio algoritmo é responsável por encontrar a configuração de operadores e/ou os valores de parâmetros para o problema sendo tratado. Alguns destes métodos utilizam informações do processo de busca e outros não. Neste trabalho é feito um estudo empírico sobre as principais abordagens aplicadas em versões autoconfiguráveis do algoritmo de Evolução diferencial que é, por sua vez, um dos atuais estado da arte na computação evolutiva. Além disso, uma nova abordagem para a configuração de operadores, baseada em técnicas de aprendizado de máquina e teoria de jogos, é proposta e avaliada.

1.1 Motivação

Idealmente, otimizar um sistema não deveria ser por si só uma tarefa difícil. Um engenheiro especialista em motores elétricos, não deveria ter que ser também um especialista em teoria da otimização, simplesmente para melhorar seus projetos. Além de ser fácil de usar um algoritmo de otimização global também deve ser confiável e robusto, ou seja, deve convergir para o ótimo em tempo razoável. Assim, um método verdadeiramente útil para otimização global deve ser fácil de implementar e de usar, confiável e rápido.

Existem vários problemas práticos que possuem algumas características peculiares como, difícil formulação matemática, espaços de busca mistos (com variáveis contínuas e discretas) e funções objetivo ruidosas, irregulares e/ou com muitas restrições, que desafiam os métodos tradicionais de otimização. Nestes tipos de problema, os AEs têm demonstrado sua utilidade sendo capazes de lidar, com maior ou menor grau de facilidade, com todas estas nuances. A principal razão que faz com que os AEs obtenham sucesso em problemas como estes é a sua flexibilidade. Contudo, contraditório como parece ser, esta flexibilidade representa uma forte limitação em seu uso pelo público em geral. A maioria dos AEs fornece aos usuários várias possibilidades de configuração para atacar as especificidades de cada problema. Embora usuários experientes possam tirar vantagem dessa variedade, um usuário comum normalmente irá falhar em configurar o algoritmo num tempo razoável. Dessa forma, para disseminar o uso dos AEs e simplificar

sua utilização para o usuário comum, parece ser essencial o oferecimento de recursos de configuração automática.

Em grande parte pelos motivos já relatados, a configuração de parâmetros e operadores é uma das áreas de pesquisa mais ativas no âmbito da computação evolutiva (ver (Lobo, Lima and Michalewicz 2007)). Normalmente essa configuração é baseada na experiência do usuário, outra possibilidade, é a utilização de métodos estatísticos derivados da área de Planejamento de Experimentos adaptados para a configuração de AEs (ver (Birattari, Stützle, Paquete and Varrentrapp 2002) e (Yuan and Gallagher 2004)). A primeira opção tem a desvantagem de exigir experiência prévia do usuário, além disso, estas duas abordagens costumam resultar em desempenhos sub-ótimos, uma vez que a estratégia mais apropriada depende do estágio do processo de otimização (Fialho, Ros, Schoenauer and Sebag 2010). No início do processo de busca é interessante ter estratégias que favoreçam a exploração do espaço, já em estágios mais avançados é interessante possuir estratégias que intensifiquem a busca nas regiões identificadas como promissoras. Por esta razão, uma prática que tem sido aplicada com sucesso em diversos contextos é a utilização de abordagens que fazem a configuração do algoritmo enquanto ele já está sendo executado (Maturana and Saubion 2008, Fialho, Costa, Schoenauer and Sebag 2008, Li, Fialho and Kwong 2011). Assim, o método de configuração pode utilizar informação do processo de busca para ajustar corretamente o algoritmo de acordo com sua necessidade no momento.

Por ser um dos atuais estado da arte em AEs, o algoritmo de Evolução Diferencial (DE) é foco de muita pesquisa, e uma série de mecanismos para sua configuração automática já foram propostos. Estes mecanismos costumam se concentrar em duas frentes: (i) na configuração automática dos parâmetros e (ii) na configuração automática de estratégias de mutação. Na frente (i) métodos comuns aplicados são: adaptação, auto-adaptação e, em alguns casos, a pura aleatorização dos valores. O problema de alocação de estratégias, representado pela frente (ii), é normalmente visto na literatura como um problema de aprendizado em ambientes dinâmicos. Um dos algoritmos mais populares para tratar esse tipo de problema é o Q-learning (Watkins 1989), contudo não parece haver ainda na literatura, trabalho que o aplique especificamente neste contexto.

Como exemplos na frente (i) temos (Brest, Greiner, Boskovic, Mernik and Zumer 2006) e (Das and Konar 2005), na frente (ii) temos (Fialho, Ros, Schoenauer and Sebag 2010) e (Gong, Fialho and Cai 2010) e ainda temos exemplos que trabalham nas duas frentes ao mesmo tempo, como (Pedrosa Silva, Lopes and Guimarães 2011) e (Qin, Huang and Suganthan 2009). Apesar de todas estas abordagens apresentarem benefício

em relação ao algoritmo DE básico, não é muito claro e parece não existir nenhum estudo que compare extensivamente o efeito de cada um destes mecanismos. Além disso, como existem vários mecanismos propostos para cada uma das frentes, pode ser que exista um efeito da interação entre eles. Efeito este que também parece ainda não ter sido mostrado.

1.2 Contribuições

Este trabalho apresenta duas contribuições importantes. A primeira é uma comparação extensa e criteriosa da maioria dos esquemas de configuração automática propostos para o DE. Com isso, pretende-se determinar efetivamente qual o efeito de cada uma destas abordagens no desempenho do algoritmo e ainda averiguar se há interação entre os métodos utilizados para a configuração de parâmetros e os métodos utilizados para a configuração de estratégias.

A segunda contribuição importante deste trabalho é a adaptação do algoritmo Q-learning para o problema de alocação de estratégias de mutação. Para tal, transformamos o algoritmo num sistema "inteligente" que aprende, via reforço, a melhor configuração a ser utilizada. Ainda em relação a este problema, um esquema de recompensa baseado em diversidade é proposto e comparado com o tradicional esquema de recompensa baseado na melhora no *fitness*.

Neste trabalho, conhecimento é gerado sobre mecanismos de configuração automática. Este conhecimento contribui para a criação de versões livres de parâmetros do DE e, poderá, futuramente, também colaborar para a criação de versões livres de parâmetros de qualquer AE.

1.3 Revisão Bibliográfica

O algoritmo de Evolução Diferencial (DE) foi proposto em 1995 para otimização com variáveis contínuas (Storn and Price 1995). Atualmente, é um algoritmo de otimização importante em sistemas mono-objetivo (Mezura-Montes, Velázquez-Reyes and Coello Coello 2006) (Price, Storn and Lampinen 2005), multiobjetivo (Babu and Jehan 2003) (Wang, Li and Yu 2010) e, mais recentemente, tem sido utilizado também em problemas combinatórios (Onwubolu and Davendra 2006) (Prado, Pedrosa Silva, Guimarães and Neto

2010) e em problemas com variáveis mistas (Freitas, Pedrosa Silva and Guimarães 2012) (Lopes, Freitas, Pedosa Silva and Guimarães 2012).

Em (Storn and Price 1997) foi exposto que os parâmetros de controle do DE são fáceis de serem fixados, contudo, Gämperle, Müller and Koumoutsakos (2002) relataram que a escolha apropriada dos parâmetros de controle era mais difícil do que se esperava. Em concordância com este estudo, Liu and Lampinen (2002) e Price, Storn and Lampinen (2005) relataram que a eficácia, eficiência e robustez do DE é sensível à configuração dos parâmetros de controle e que o conjunto mais adequado destes parâmetros depende da função que se quer otimizar e dos requerimentos de tempo e precisão do problema.

Devido à dificuldade na fixação de parâmetros, foi proposta por Liu and Lampinen (2005) uma versão do DE em que os parâmetros de controle de mutação e cruzamento são adaptados utilizando-se um mecanismo baseado em lógica nebulosa. Com o mesmo propósito, em (Epitropakis, Plagianakos and Vrahatis 2009) uma população de parâmetros evolui em paralelo com a população de soluções e seu *fitness* é calculado de acordo com a qualidade das soluções geradas por eles. Dessa forma, parâmetros que geram soluções melhores possuem mais chance de sobreviver e reproduzir. Nestas duas abordagens, no geral, os resultados foram superiores aos do DE convencional no conjunto de funções testadas.

Uma outra família de métodos de configuração automática de parâmetros do DE é inspirada na abordagem conhecida como autoadaptação. Esta abordagem é proveniente dos métodos conhecidos como Estratégias Evolutivas (Schwefel 1981), que são um tipo de AE que incorpora os parâmetros à representação do indivíduo, e fazem com que eles também sofram ação dos operadores genéticos. Dessa forma, os parâmetros são ajustados automaticamente pelo próprio algoritmo. Em (Brest, Greiner, Boskovic, Mernik and Zumer 2006) e (Salman, Engelbrecht and Omran 2006) são propostas versões do DE com este tipo de mecanismo para configurar os parâmetros. Todas estas versões se mostraram superiores ao DE convencional de parâmetros fixos na maior parte do conjunto de testes realizados. Na mesma linha, Teo (2006) apresentou uma tentativa de configurar automaticamente o tamanho da população, em adição à configuração das taxas de mutação e cruzamento.

Em (Qin, Huang and Suganthan 2009) é relatado que, além da correta configuração de parâmetros, o desempenho do DE convencional também depende fortemente da escolha dos operadores de variação utilizados. Existem na literatura vários trabalhos relacionados a métodos de Seleção Automática de Operadores, ou Seleção Adaptativa

de Operadores (SAO), para AEs. Na maioria destes trabalhos o processo é divido em duas partes:

- Atribuição de crédito: responsável por atribuir uma recompensa numérica a um determinado operador, baseado em alguma medida de qualidade das soluções geradas por ele.
- Mecanismo de alocação de operador: responsável por determinar o operador a ser aplicado, baseado nas recompensas recebidas por ele.

O primeiro método de SAO foi proposto por Davis (1989). Nesse trabalho, as recompensas eram baseadas na frequência com que os operadores geravam soluções melhores que a melhor solução da população. A mesma ideia pode ser vista em (Julstrom 1995), a diferença é que neste trabalho o operador recebe uma recompensa se os descendentes gerados são melhores que os pais e não melhores que o melhor indivíduo. Em ambos os trabalhos o mecanismo de alocação de operadores é similar ao método de Casamento de Probabilidades (do inglês, Probability Matching), no qual a probabilidade de alocação de um determinado operador é proporcional às recompensas recebidas por ele. Bons resultados são relatados nos dois trabalhos quando os métodos propostos foram comparados com versões estáticas dos algoritmos.

Em (Whitacre, Pham and Sarker 2006) é apresentada uma abordagem de atribuição de crédito baseada em um método estatístico de detecção de *outliers*. Os resultados mostraram que algoritmos com este mecanismo de alocação apresentam desempenho superior aos baseados em desempenho médio. Devido à complexidade de aplicação de métodos de detecção de *outliers*, em (Fialho, Costa, Schoenauer and Sebag 2008) é apresentada uma abordagem de atribuição de créditos baseada em valores extremos. Neste trabalho, armazena-se em um arquivo as recompensas calculadas das últimas gerações. A recompensa a ser efetivamente atribuída ao operador será a maior do arquivo. O objetivo dessa estratégia é emular o efeito da técnica de detecção de *outliers*, favorecendo operadores que produzem soluções marcadamente boas (embora raras) em detrimento de operadores que possuam apenas bom desempenho médio. Em (Maturana and Saubion 2008), (Maturana, Fialho, Saubion, Schoenauer and Sebag 2009) e (Fialho, Schoenauer and Sebag 2009) medidas de diversidade são acopladas às medidas de melhora de *fitness* para a atribuição de crédito.

Especificamente para o DE, em (Qin and Suganthan 2005) além dos parâmetros de controle do algoritmo, as probabilidades de aplicação de variações do operador de

mutação são adaptadas de acordo com o número de vezes nas quais estes geraram soluções melhores que os pais. Primeiramente foi utilizada uma versão com duas estratégias de mutação diferentes e, posteriormente, em (Qin, Huang and Suganthan 2009), foram utilizadas quatro estratégias. Em ambos os casos o mecanismo de alocação é similar ao casamento de probabilidades.

Pant, Ali and Abraham (2009) apresentaram um SAO baseado em teoria de jogos. Nesta abordagem os indivíduos são considerados como jogadores aplicando operadores de mutação diversos para gerar descendentes. Diferentemente dos outros métodos descritos, neste caso não existe atribuição de crédito aos operadores. Cada solução possui um vetor de probabilidades de aplicação das estratégias de mutação, e estas probabilidades são incrementadas ou decrementadas diretamente, dependendo da qualidade do descendente em relação ao pai. Os resultados obtidos foram competitivos em relação às abordagens comparadas, contudo, o método introduz um novo parâmetro ao qual seu desempenho é bastante sensível, o que dificulta sua utilização.

Em (Gong, Fialho and Cai 2010) é utilizado novamente um método de casamento de probabilidades, porém, agora com uma atribuição de crédito baseada em melhora no *fitness*. Em (Li, Fialho and Kwong 2011) uma abordagem parecida é apresentada para uma versão multiobjetivo do DE, porém, neste caso os conceitos de dominância Pareto são incorporados na atribuição de crédito.

Em (Fialho, Ros, Schoenauer and Sebag 2010) é proposto um mecanismo de alocação baseado em *Multi-armed bandits* para o DE. Nele a estratégia de mutação é selecionada de forma determinística de acordo com a recompensa (baseada em melhora no *fitness*) recebida. Este método foi comparado com a abordagem baseada em casamento de probabilidades, com uma abordagem na qual as estratégias de mutação eram escolhidas de forma aleatória e com uma abordagem baseada em busca adaptativa[2]. Apesar do método proposto ter apresentado menor tempo de execução, não foi detectada diferença significativa em relação à qualidade das soluções obtidas.

Por fim, em (Wang, Cai and Zhang 2011) é apresentada uma versão do DE com múltiplas estratégias de mutação e um conjunto fixo de pares de parâmetros de forma a prover formas diversificadas de exploração ao algoritmo. Para cada indivíduo todos os operadores de mutação são aplicados na geração de descendentes dos quais apenas

[2]Assim como no método de Casamento de Probabilidades, a Busca Adaptativa adapta probabilidades de utilização de estratégias, baseada nas recompensas recebidas. A diferença é que no segundo método as probabilidades não são diretamente proporcionais às recompensas. Nele há uma maior inclinação à escolha da estratégia de maior recompensa do momento.

o melhor sobrevive. Os pares de parâmetros são escolhidos aleatoriamente dentro do conjunto pré-selecionado. Esta abordagem apresentou resultados significativamente melhores que outras abordagens adaptativas e convencionais do DE no conjunto de funções de teste utilizado.

1.4 Organização do Trabalho

O restante deste trabalho está organizado da seguinte forma: No **Capítulo 2 - Evolução Diferencial e Configuração Automática**, é apresentado o algoritmo de Evolução Diferencial Básico. Seus principais operadores e os parâmetros que controlam estes operadores são discutidos e analisados. Em seguida é feita uma revisão detalhada dos métodos de configuração automática, tanto de parâmetros quanto de estratégias de variação e é proposto o algoritmo de seleção baseado em Q-learning. Por fim, um arcabouço genérico para versões auto-configuráveis do DE é descrito. No **Capítulo 3 - Experimentos Computacionais**, é descrito todo o planejamento experimental utilizado e os resultados computacionais são apresentados e discutidos. Por fim, no **Capítulo 4 - Conclusões**, são apresentadas as conclusões do trabalho.

Capítulo 2

Evolução Diferencial e Configuração Automática

2.1 Evolução Diferencial

Problemas de otimização são onipresentes na ciência e na engenharia e por isso a importância de métodos para tratá-los. Os usuários de uma técnica de otimização desejam que ela preencha três requerimentos básicos:

1. o método deve achar o ótimo global, independente do estado inicial do processo de busca;
2. a convergência deve ser rápida; e
3. o método deve ter o mínimo de parâmetros de controle, para que ele seja fácil de usar.

Na busca por uma técnica que fosse de encontro aos critérios citados anteriormente, em (Storn and Price 1995) foi proposto o método conhecido como Evolução Diferencial (DE - Differential Evolution). Além de ser simples e poderoso, por ser baseado em populações, o DE é inerentemente paralelo, possibilitando sua implementação em conjuntos de computadores e processadores (Arabas, Maitre and Collet 2012).

O sucesso do DE deve-se, principalmente, ao seu poderoso e simples mecanismo de busca, que usa a diferença entre dois vetores, escolhidos aleatoriamente dentre as soluções

candidatas, para produzir novas soluções. À medida que a população evolui, a direção e o tamanho dos passos dados pelo algoritmo mudam, ajustando-se de acordo com a distribuição da população no espaço de busca. O DE usa uma abordagem gulosa e, ao mesmo tempo, estocástica para solucionar o problema de otimização. Ele combina operadores aritméticos simples com as operações clássicas de cruzamento, mutação e seleção, de modo que a população inicialmente aleatória possa evoluir para uma população com soluções de alta qualidade.

Apesar do método de evolução diferencial ser classificado como um algoritmo evolutivo, e como já pode-se perceber, se enquadra no esquema geral de algoritmos evolutivos (ver Fig.1.2), seu operador de mutação não tem inspiração em nenhum processo natural. Este operador, conhecido como *mutação diferencial*, se sustenta em argumentos matemáticos e heurísticos, que indicam sua adequação para a otimização de funções, como poderemos ver a seguir.

2.1.1 Algoritmo de Evolução Diferencial

Consideremos a versão irrestrita do problema genérico de otimização, descrito pela Eq.1.1, e $\mathcal{U}_{[a,b]}$ a amostragem de uma variável aleatória com distribuição uniforme entre a e b. De acordo com sua definição original, apresentada em (Storn and Price 1995), o DE consiste no seguinte.

Seja uma população de soluções candidatas, geradas aleatoriamente no espaço de busca, representada por $X_t = \{\mathbf{x}_{t,i}; i = 1, 2, ..., NP\}$. t é o índice da geração corrente, i é o índice do indivíduo na população e NP é número de indivíduos na população. Cada indivíduo na população corrente é representado por um vetor coluna:

$$\mathbf{x}_{t,i} = \begin{bmatrix} \mathbf{x}_{t,i,1} \\ \mathbf{x}_{t,i,2} \\ \vdots \\ \mathbf{x}_{t,i,n} \end{bmatrix} \tag{2.1}$$

onde o terceiro índice indica uma entre as n variáveis do problema de otimização.

O mecanismo de busca do DE utiliza a diferença entre pares de vetores escolhidos

aleatoriamente dentre as soluções candidatas da população. O vetor resultante dessa diferença é adicionado a uma terceira solução, também escolhida aleatoriamente. A equação a seguir ilustra este procedimento:

$$\mathbf{v}_{t,i} = \mathbf{x}_{t,r1} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) \quad r1 \neq r2 \neq r3 \neq i \tag{2.2}$$

$\mathbf{v}_{t,i}$ representa a i-ésima solução mutante e F é o fator de escala aplicado ao vetor diferencial, parâmetro do algoritmo DE. O vetor $\mathbf{x}_{t,r1}$, ao qual é aplicada a mutação diferencial, é denominado *vetor base*.

Após a realização deste procedimento é obtida a população mutante $V_t = \{\mathbf{v}_{t,i}; i = 1, 2, ..., NP\}$. V_t então entrará na fase de recombinação com a população corrente X_t, produzindo uma população de descendentes, chamados indivíduos teste, U_t. Originalmente é empregada a recombinação binomial (ou uniforme) com probabilidade $CR \in [0, 1]$ ilustrada pela equação a seguir:

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{v}_{t,i,j}, & \text{se } \mathcal{U}_{[0,1]} \leq CR \ \vee \ j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.3}$$

em que $\delta_i \in 1, \cdots, n$ é um índice aleatório sorteado para o vetor teste i e serve para garantir que pelo menos uma variável da solução teste seja herdada do indivíduo mutante. Notem que o parâmetro CR controla a fração de valores em $\mathbf{u}_{t,i}$ que são copiados do vetor mutante $\mathbf{v}_{t,i}$. Se $CR = 1$ o vetor teste será idêntico ao vetor mutante.

A Figura 2.1 ilustra a geração de um vetor mutante e as possíveis soluções teste obtidas após a recombinação, indicadas por ■. Note que pelo menos uma coordenada $j = \delta_i$ será herdada do vetor mutante, de forma a garantir $\mathbf{u}_{t,i} \neq \mathbf{x}_{t,i}$. Pode-se observar que as soluções mutantes são geradas a partir de uma perturbação em algum indivíduo da população. A direção e o tamanho dessa perturbação são definidos pela diferença entre as soluções $\mathbf{x}_{t,r2}$ e $\mathbf{x}_{t,r3}$, portanto dependem das posições relativas destes indivíduos no espaço de busca.

Por fim, a solução teste $\mathbf{u}_{t,i}$ é avaliada calculando-se seu valor de função objetivo. Cada solução teste $\mathbf{u}_{t,i}$ é comparada com seu correspondente $\mathbf{x}_{t,i}$ da população corrente

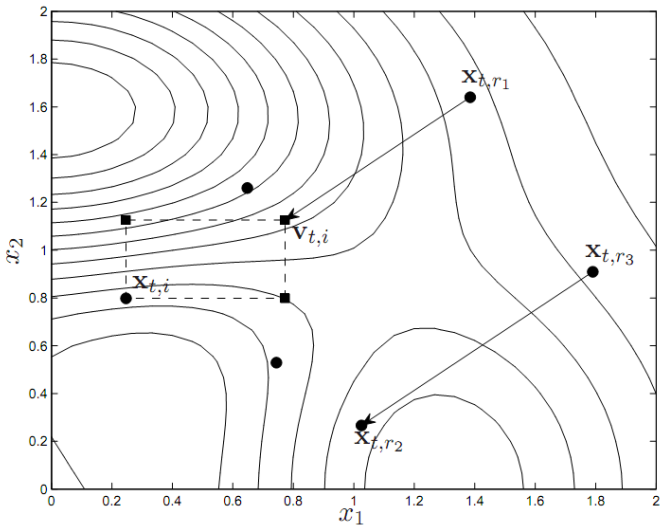


Figura 2.1: Ilustração do procedimento de geração de uma solução mutante do Algoritmo de Evolução Diferencial(Guimarães 2009)

e a melhor entre as duas passa para a população da próxima geração, a pior dentre as duas é eliminada. Este procedimento é ilustrado pela equação a seguir:

$$\mathbf{x}_{t+1,i} = \begin{cases} \mathbf{u}_{t,i}, & \text{se } f(\mathbf{u}_{t,i}) < f(\mathbf{x}_{t,i}) \\ \mathbf{x}_{t,i}, & \text{caso contrário} \end{cases} \tag{2.4}$$

Podemos observar ainda que as equações (2.2) e (2.3), podem ser escritas de forma compacta como segue:

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,r1,j} + F(\mathbf{x}_{t,r2,j} - \mathbf{x}_{t_r3,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \ \vee \ j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.5}$$

logo, as equações (2.5) e (2.4) descrevem todas as operações do algoritmo de evolução diferencial em sua versão original. O pseudocódigo do DE é apresentado no Algoritmo1.

Algoritmo 1: Algoritmo de Evolução Diferencial

$t \leftarrow 1$
Inicializar População $X_t = \{\mathbf{x}_{t,i}; i = 1, 2, ..., NP\}$
Avalia X_t
while not condição_de_parada **do**
  for $i = 1$ to NP **do**
    Selecione Aleatoriamente $r_1, r_2, r_3 \in 1, \cdots, NP$
    Selecione Aleatoriamente $\delta_i \in 1, \cdots, n$
    for $j = 1$ to n **do**
      if $\mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i$ **then**
        $\mathbf{u}_{t,i,j} = \mathbf{x}_{t,r1} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t_r3})$
      else
        $\mathbf{u}_{t,i,j} = \mathbf{x}_{t,i,j}$
      end if
    end for
  end for
  for $i = 1$ to NP **do**
    if $f(\mathbf{u}_{t,i}) < f(\mathbf{x}_{t,i})$ **then**
      $\mathbf{x}_{t+1,i} \leftarrow \mathbf{u}_{t,i}$
    else
      $\mathbf{x}_{t+1,i} \leftarrow \mathbf{x}_{t,i}$
    end if
  end for
  $t \leftarrow t + 1$
end while

2.1.2 Comportamento da Mutação Diferencial

O princípio básico de funcionamento do algoritmo de evolução diferencial é perturbar soluções da população corrente gerando vetores mutantes pelo procedimento descrito pela Eq.2.2. Essas perturbações, também chamadas de *vetores diferença*, são proporcionais à diferença entre pares de soluções escolhidas aleatoriamente na população. Numa população de NP vetores distintos haverá $NP \times (NP - 1)$ vetores diferença não nulos. A Figura 2.2 retrata uma população arbitrária de 15 vetores e os vetores diferença não nulos que eles geram.

Nas Figuras 2.2 à 2.4 são ilustrados alguns estágios da execução do DE para a otimização de uma função quadrática, cujas curvas de nível correspondem a elipsoides rotacionados de $\pi/4$ no sentido anti-horário em relação aos eixos coordenados. Além disso, um dos eixos desse elipsoide é maior do que o outro, tornando o elipsoide "alongado" numa dada direção.

A Figura 2.2-(a) mostra a configuração inicial da população X_t no espaço de busca. Neste estágio, as soluções estão distribuídas aleatoriamente com distribuição uniforme na região retangular correspondentes aos limites inferiores e superiores de cada variável. A Figura 2.2-(b) mostra que a distribuição de vetores diferença correspondentes não possui nenhum tipo de polarização, e é formada por vetores de diversos tamanhos.

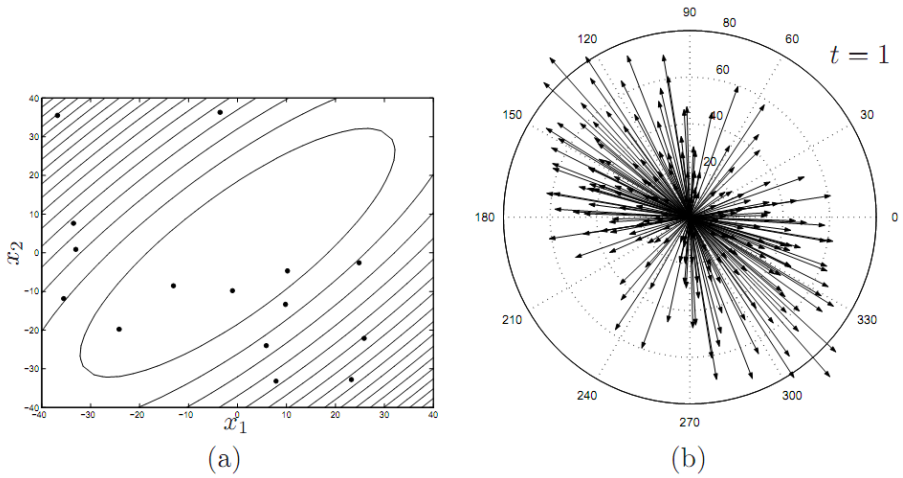


Figura 2.2: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 1$ (b) Distribuição dos vetores diferenciais na geração $t = 1$ (Guimarães 2009)

À medida que o processo de otimização continua, novas soluções vão sendo geradas.

Soluções ruins, com maior valor de função objetivo, vão sendo substituídas por soluções mais próximas do mínimo, alterando a distribuição das soluções da população X_t.

A Figura 2.3-(a) ilustra a distribuição espacial 10 gerações após a distribuição inicial. Observe que neste estágio a distribuição dos vetores diferença correspondente, mostrada na Figura 2.3-(b), se alinha à forma da função, possuindo direções mais favoráveis à sua minimização. Outro fator importante a ser observado é que o tamanho dos vetores, que darão os tamanhos das perturbações a serem aplicadas aos demais indivíduos, diminuíram devido à aglomeração dos indivíduos em torno do ponto de mínimo.

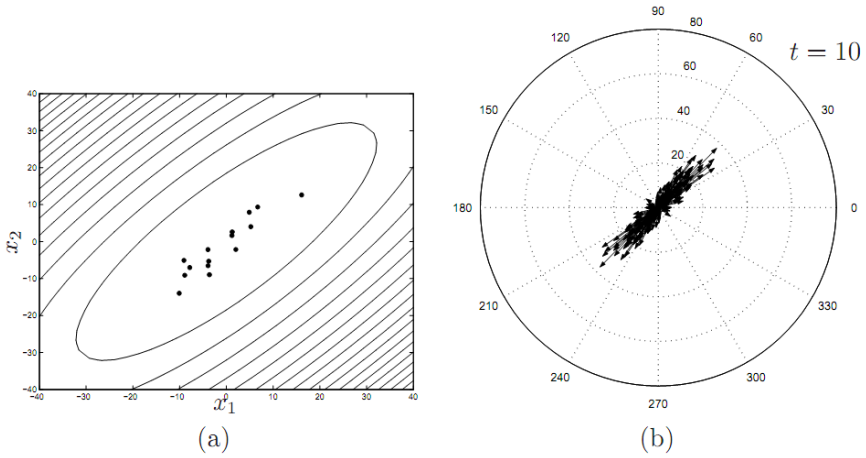


Figura 2.3: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 10$ (b) Distribuição dos vetores diferenciais na geração $t = 10$ (Guimarães 2009)

A Figura 2.4 ilustra a distribuição espacial da população e a distribuição dos vetores diferenciais 20 gerações após a distribuição inicial. A população está agora mais próxima do ponto de mínimo e ocupa um volume reduzido em relação à distribuição espacial em $t = 1$. A distribuição dos vetores diferenciais continua alinhada com os elipsoides que formam as curvas de nível da função, porém, os tamanhos desses vetores estão bem reduzidos, favorecendo a intensificação da busca. Nesse momento, o algoritmo converge para o ótimo global.

Neste exemplo podemos observar que a distribuição dos vetores diferença vai depender da distribuição da população. E, esta por sua vez, será diferente dependendo da função objetivo e do estágio do processo de otimização.

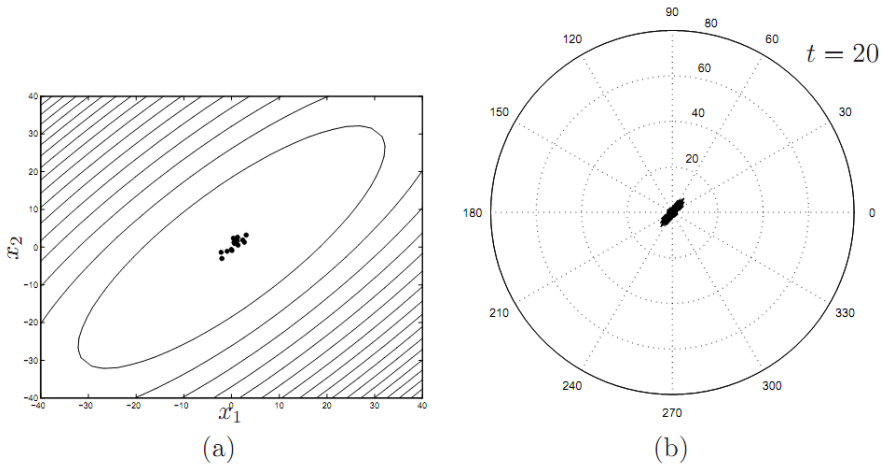


Figura 2.4: Função-objetivo quadrática. (a) Distribuição espacial da população na geração $t = 20$ (b) Distribuição dos vetores diferenciais na geração $t = 20$ (Guimarães 2009)

Fator de escala F

No DE, grande parte da responsabilidade de adaptar o tamanho de passo das mutações é deixada para a própria distribuição das soluções. O fator de escala F (ver Eq.2.2) só afeta o tamanho de passo relativo uma vez que a distribuição dos vetores diferença é por si só adaptativa. Assim, F pode ser mantido constante durante a otimização sem comprometer a habilidade do algoritmo em gerar passos de tamanho adequado. Contudo, variar F durante o processo de busca pode oferecer benefícios potenciais.

Fornecer diferentes valores de F ao algoritmo amplia o número de possíveis vetores diferença. Isso é particularmente útil pois se a distribuição da mutação, dada pela distribuição dos vetores diferença, não tiver diversidade suficiente o algoritmo pode cair em uma condição indesejável de estagnação[1]. Quando em estagnação, o DE não consegue encontrar soluções melhoradas pois nenhuma combinação vetor e vetor diferença leva a uma solução melhor. Esta condição foi estudada por Lampinen and Zelinka (2000) em um caso hipotético e como relatado em (Price, Storn and Lampinen 2005) nenhuma tentativa subsequente de induzir estagnação no DE com funções de teste foi bem sucedida. Todavia, variar F é uma maneira de aumentar o conjunto de possíveis vetores teste e minimizar o risco de estagnação sem aumentar o tamanho da população.

[1]No DE, estagnação é o fenômeno que ocorre quando: (i) o algoritmo falha em encontrar soluções melhore e; (ii) o algoritmo não converge para uma solução (nem sub-ótima); mesmo quando a diversidade da população continua alta. (Ver (Lampinen and Zelinka 2000))

Limites de F

O intervalo estabelecido na literatura para F é $(0, 1)$. O limite superior 1 foi derivado empiricamente, no sentido de que nenhuma função otimizada com sucesso requereu $F > 1$ (Price, Storn and Lampinen 2005). Valores de F maiores do que 1 são usados raramente, o que não significa que soluções não possam ser encontradas com estes valores. Quando $F = 1$ entretanto, algumas combinações de vetores se tornam indistinguíveis como mostra a Eq.2.6. Neste caso o número de mutantes cai pela metade o que pode levar o algoritmo a ter problemas para convergir.

$$\mathbf{x}_{t,r0} + \mathbf{x}_{t,r1} - \mathbf{x}_{t,r2} = \begin{cases} \mathbf{x}_{t,r0} + F \cdot (\mathbf{x}_{t,r1} - \mathbf{x}_{t,r2}) \\ \mathbf{x}_{t,r1} + F \cdot (\mathbf{x}_{t,r0} - \mathbf{x}_{t,r2}) \end{cases} \quad \text{onde } F = 1 \tag{2.6}$$

Zaharie (2002) demonstrou a existência de um limite inferior para F. Se $F < \sqrt{(1 - p_m/2)/NP}$, onde p_m é a probabilidade de uma componente do vetor teste $\mathbf{u}_{t,i}$ ser herdado do vetor mutante $\mathbf{v}_{t,i}$, então a população perde diversidade mesmo sem pressão seletiva. Esse valor crítico depende do tamanho da população NP e da probabilidade de cruzamento CR. Apesar de F precisar ter magnitude suficiente para evitar a estagnação, em (Chakraborti, Misra, Bhatt, Barman and Prasad 2001) obteve-se sucesso na resolução de alguns problemas com $F = 0.0001$.

2.1.3 Comportamento da Recombinação

Nos Algoritmos Evolutivos, em geral, o operador de recombinação combina as características dos pais. No caso do DE a mutação já é responsável por combinar informação de vários indivíduos, neste caso o operador de recombinação age apenas misturando componentes da solução corrente $\mathbf{x}_{t,i}$ com o vetor mutante $\mathbf{u}_{t,i}$. A Figura 2.5 ilustra o procedimento de recombinação descrito pela Eq.2.3.

Como podemos observar, o papel do CR está mais ligado a uma probabilidade de mutação, uma vez que ele influencia no número de componentes que serão herdados da solução mutante. Como relatado em (Price, Storn and Lampinen 2005), é fácil perceber que valores de CR de baixa magnitude ($0 \leq CR \leq 0.2$) são interessantes para a otimização de funções separáveis, pois cada variável pode ser otimizada independentemente.

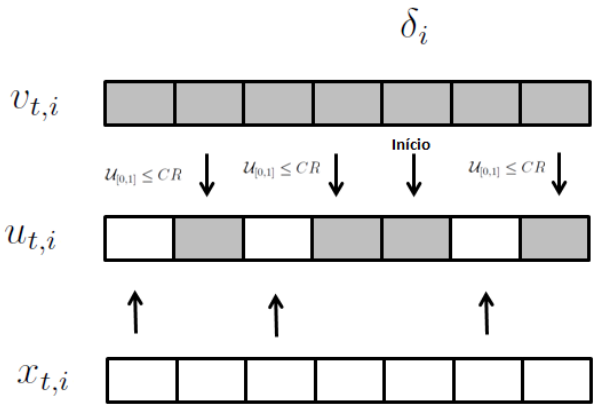


Figura 2.5: Recombinação Binomial

Por outro lado, para problemas não separáveis valores altos de CR ($0.9 \leq CR \leq 1$) já são mais recomendados, pois uma vez que há uma correlação entre as variáveis é interessante que elas sejam otimizadas simultaneamente.

O principal problema deste tipo de recombinação é o fato dele não ser invariante à rotação, tornando o DE menos eficiente em funções rotacionadas. Por outro lado eliminá-la completamente leva o DE a ter um baixo desempenho em problemas multimodais (Zaharie 2009). Para eliminar este tipo de problema, alguns trabalhos, e.g. (Qin, Huang and Suganthan 2009) e (Wang, Cai and Zhang 2011) utilizam versões do DE sem recombinação binomial. Nestas abordagens é utilizada a recombinação aritmética, que faz uma combinação linear das soluções, no lugar do operador tradicional. Neste procedimento se quiséssemos recombinar as soluções $\mathbf{x}_{t,r0}$ e $\mathbf{x}_{t,r1}$ geraríamos a solução $\mathbf{w}_{t,i}$ de acordo com a Eq.2.7.

$$\mathbf{w}_{t,i} = \mathbf{x}_{t,r0} + Z_i(\mathbf{x}_{t,r1} - \mathbf{x}_{t,r0}) \tag{2.7}$$

onde $Z \in (0, 1)$ é o coeficiente de recombinação que determinará para qual solução, $\mathbf{x}_{t,r1}$ ou $\mathbf{x}_{t,r0}$, o novo indivíduo irá tender. Na seção a seguir são apresentadas e discutidas algumas variantes importantes do DE.

2.1.4 Variações da Estratégia de Geração de Vetores Teste

Na literatura relativa ao DE, existem várias versões das estratégias de geração de vetores teste. A nomeclatura destas versões, segue o seguinte formato:

DE/base/n/rec

onde,

- base - representa o vetor ao qual será aplicada a mutação;
- n - se refere ao número de vetores diferença utilizados na mutação; e
- rec - se refere ao tipo de recombinação utilizado. Quando utiliza-se a recombinação aritmética esta parte é suprimida do nome.

O algoritmo DE empregando diferentes estratégias de variação, normalmente, possui desempenho distinto dependendo do estágio da evolução (Qin, Huang and Suganthan 2009). Mezura-Montes, Velázquez-Reyes and Coello Coello (2006) apresentam uma comparação empírica dentre diversas variações destes operadores. Algumas estratégias favorecem a exploração, outras favorecem a intensificação da busca e cada uma funciona melhor num determinado tipo de problema. As principais variantes podem ser divididas em três grupos principais:

1. Estratégias Baseadas no Melhor Indivíduo Corrente (**best**)

   - **best/1/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,best,j} + F(\mathbf{x}_{t,r2,j} - \mathbf{x}_{t,r3,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.8}$$

   - **current-to-best/2/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,i,j} + F(\mathbf{x}_{t,best,j} - \mathbf{x}_{t,i,j}) + F(\mathbf{x}_{t,r1,j} - \mathbf{x}_{t,r2,j}) + F(\mathbf{x}_{t,r3,j} - \mathbf{x}_{t,r4,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.9}$$

2. Estratégias Baseadas em Vetor Aleatório

- **rand/1/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,r1,j} + F(\mathbf{x}_{t,r2,j} - \mathbf{x}_{t,r3,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.10}$$

- **rand/2/bin**

$$\mathbf{u}_{t,i,j} = \begin{cases} \mathbf{x}_{t,r1,j} + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) + F(\mathbf{x}_{t,r4} - \mathbf{x}_{t,r5}), & \text{se } \mathcal{U}_{[0,1]} \leq CR \vee j = \delta_i \\ \mathbf{x}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.11}$$

3. Estratégias Invariantes a Rotação

- **current-to-rand/1**

$$\mathbf{u}_{t,i} = \mathbf{x}_{t,i} + Z(\mathbf{x}_{t,r1} - \mathbf{x}_{t,i}) + F(\mathbf{x}_{t,r2} - \mathbf{x}_{t,r3}) \tag{2.12}$$

As estratégias baseadas na melhor solução corrente, normalmente, possuem rápida convergência e desempenho superior quando tratam problemas unimodais. Contudo, elas também possuem uma maior tendência a ficarem presas em ótimos locais levando à convergência prematura em problemas multi-modais. As estratégias baseadas em vetores aleatórios são comumente conhecidas por possuir alto poder de exploração e velocidade de convergência relativamente baixa. As estratégias invariantes a rotação fazem uma recombinação aritmética ao invés da recombinação binomial clássica. Este tipo de estratégia é interessante pois normalmente é pouco provável que se saiba a priori a orientação da função objetivo. É importante notar que este tipo de estratégia apenas garante a independência do sistema de coordenadas, contudo, todas as outras estratégias se tornam invariantes a rotação se $CR = 1$.

Outro ponto importante sobre estas variantes é o emprego de dois vetores diferença no lugar de apenas um, como proposto originalmente. Em (Zhang and Xie 2003) é demostrado empiricamente, num contexto de hibridização de DE com PSO, que a distribuição estatística das perturbações com apenas um vetor diferença possui forma de triângulo, enquanto que com dois vetores diferença a distribuição possui forma de sino. Esta distribuição é geralmente considerada como um modo de perturbação melhor (Qin, Huang and Suganthan 2009).

2.2 Configuração Automática

Como pudemos observar na seção anterior, o DE básico, apresentado pelo Algoritmo 1, é apenas um arcabouço que pode ser modificado de uma série de maneiras de forma a atacar as especificidades de diversos tipos de problema. Entretanto, esse grande número de possibilidades pode representar uma limitação do seu uso por usuários inexperientes. Mesmo em se tratando de usuários experientes, a configuração correta do algoritmo demandaria um conhecimento profundo do problema em mãos. Algumas informações importantes para a correta configuração do método são, por exemplo, se o problema é separável ou não, se está rotacionado em relação aos eixos coordenados, e se é uni ou multi modal. Estas informações, entretanto, raramente estão disponíveis, principalmente quando estamos tratando problemas reais, para os quais o modelo é tipo "caixa preta".

Neste contexto, o que resta ao usuário comum é a utilização de uma configuração padrão que pode levar a um desempenho pobre do algoritmo. Assim, para que o DE se firme como ferramenta de otimização geral, alguma capacidade de configuração automática deve ser fornecida. Por este motivo existe um grande número de trabalhos relacionados a este tema. Nestes trabalhos o problema de configuração automática do DE é atacado em duas frentes. A primeira trata da configuração de parâmetros e a segunda da configuração de estratégias de variação (recombinação + mutação). Nas próximas seções estas frentes serão apresentadas, e as principais abordagens presentes na literatura para cada uma delas serão descritas.

2.2.1 Configuração de Parâmetros

A configuração de parâmetros pode ser feita de inúmeras formas. Eiben and Schut (2008) dividem estas formas em dois grandes grupos, conhecidos como:

1. Ajuste de Parâmetros: Nesta forma tenta-se encontrar bons valores de parâmetros antes da execução do algoritmo. O método é ajustado com estes valores, os quais se manterão fixos durante toda a execução;
2. Controle de Parâmetros: Nesta forma os parâmetros são modificados dinamicamente durante a execução do algoritmo.

Estas duas formas podem ser feitas de forma automática, contudo, como relatado por Fialho, Ros, Schoenauer and Sebag (2010), uma abordagem que resulta numa estratégia

fixa pode resultar num desempenho sub-ótimo uma vez que a estratégia mais apropriada depende do estágio do processo de otimização. Por este motivo o presente trabalho se concentrará apenas na segunda forma, que ainda segundo Eiben and Schut (2008) pode ser dividida em três classes:

- Determinístico: Ocorre quando o valor do parâmetro é alterado por alguma regra determinística e, em geral, heurística que é ativada em momentos predeterminados e causa uma mudança também predeterminada sem a utilização de nenhuma informação do processo de busca.

- Adaptativo: Ocorre quando alguma informação do processo de busca é utilizada para realimentar o mecanismo de variação de parâmetros. Apesar das ações do EA poderem determinar se os valores adaptados persistirão, é importante notar que o mecanismo de controle é alimentado externamente e não faz parte do ciclo evolutivo.

- Autoadaptativo: Ocorre quando os parâmetros a serem configurados são codificados na representação do indivíduo e passam a sofrer ação dos operadores de variação (mutação e recombinação). Quanto melhores forem os valores dos parâmetros codificados, melhores as soluções geradas por eles. Estas soluções, por sua vez, possuem uma maior chance de sobreviver e produzir descendentes, propagando estes bons valores de parâmetro. Neste caso, a informação utilizada para variação e seleção dos parâmetros é fornecida indiretamente pelo próprio processo evolutivo.

A Figura 2.6 sintetiza a taxonomia apresentada.

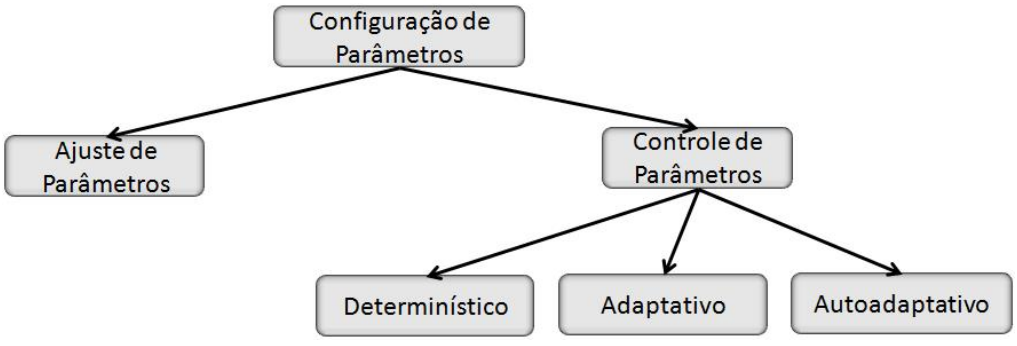


Figura 2.6: Taxonomia da Configuração de Parâmetros.

A seguir serão apresentadas as principais técnicas de controle de parâmetros presentes na literatura relacionada ao DE.

Aleatorização

Uma das formas mais simples de retirar do usuário a responsabilidade de configurar os parâmetros de controle do algoritmo é a aleatorização dos mesmos. Seus benefícios são relatados em (Price, Storn and Lampinen 2005) e dentre eles estão a redução do número de avaliações de função para a convergência e a minimização do risco de estagnação.

Esta técnica não utiliza nenhuma informação do processo de busca e consiste basicamente em transformar os parâmetros em variáveis aleatórias. A Eq.2.13 apresenta um mecanismo bastante comum na literatura para a realização da aleatorização (ver. (Brest, Boskovic, Greiner, Zumer and Maucec 2007),(Brest, Greiner, Boskovic, Mernik and Zumer 2006),(Zamuda, Brest, Boskovic and Zumer 2009),(Liu, Shi and Chen 2011)).

$$P'_{g,i} = \mathcal{U}_{[a,b]} \text{ , se } \mathcal{U}_{[0,1]} \leq \tau \tag{2.13}$$

P' é o parâmetro gerado por este procedimento e tipicamente representa F e CR. $\mathcal{U}_{[a,b]}$ representa a amostragem de uma variável aleatória com distribuição uniforme no intervalo $[a, b]$ e $\tau \in [0, 1]$ representa a chance do parâmetro ser re-amostrado. Considerando que a geração de $\mathcal{U}_{[0,1]}$ tem custo $O(1)$, o custo de geração de P' por esta técnica também é $O(1)$, mesmo que seu valor não seja alterado.

Outra abordagem comum é a amostragem com distribuição gaussiana descrita pela Eq.2.14.

$$P'_{g,i} = \mathcal{N}_{[a,b]} \tag{2.14}$$

onde, $\mathcal{N}_{[a,b]}$ é a amostragem de uma variável aleatória com distribuição gaussiana com média a e desvio padrão b. Este procedimento pode ser visto em (Qin, Huang and Suganthan 2009), (Price, Storn and Lampinen 2005) e (Zaharie 2002).

Aleatorização com Adaptação

Este tipo de abordagem tem uma grande semelhança com a abordagem anterior, porém, agora faz-se uso de informação do processo de busca para a geração de novos parâmetros. A Eq. 2.15 descreve a geração de novos parâmetros por este processo.

$$P'_{g,i} = \mathcal{N}_{[P_m,b]} \tag{2.15}$$

A principal diferença desta abordagem em relação à anterior é a presença de um mecanismo de "aprendizado" na forma da adaptação de um parâmetro que controla a amostragem, no caso, a média da distribuição gaussiana (P_m). Essa abordagem pode ser vista em (Qin, Huang and Suganthan 2009), (Qin and Suganthan 2005), (Zhang and Sanderson 2007) e (Zhang and Sanderson 2009).

Nos dois primeiros trabalhos, durante Lg gerações, é feita uma lista (L_P) dos valores de parâmetros (no caso, somente valores de CR) que geraram descendentes melhores que os pais. Então a cada Lg gerações a mediana desta lista se torna o novo P_m como descrito pela Eq.2.16. Considerando que o custo de se fazer $\mathcal{N}_{[a,b]}$ é $O(1)$ e que ao final de Lp gerações teremos uma lista de tamanho l, o custo de cálculo da mediana é $O(l\log_2 l)$ e portanto o custo de geração de P' é também $O(l\log_2 l)$.

$$P_m = \text{mediana}(L_P) \tag{2.16}$$

Nos dois últimos trabalhos novamente é feita uma lista com os valores de parâmetros que geraram descendentes melhores que os pais, porém, agora existe uma lista para F, L_F e uma lista para CR, L_{CR}. F_m é calculado como descrito pela Eq.2.17 e CR_m é calculado como descrito pela Eq.2.18

$$F_m(g) = (1-c)F_m(g-1) + c(\text{lehmer}(L_F)) \tag{2.17}$$

$$CR_m(g) = (1 - c)CR_m(g - 1) + c(\text{media}(L_{CR})) \tag{2.18}$$

onde lehmer($\cdot$) é a média de Lehmer[2], media($\cdot$) é a média aritmética e c é um parâmetro do algoritmo. Como podemos observar o custo de geração de um novo valor de parâmetro por esta abordagem é $O(l)$, que é justamente o custo de cálculo das médias de um conjunto de tamanho l.

Conjunto Fixo de Estratégias

Esta é uma abordagem muito simples proposta recentemente por Wang, Cai and Zhang (2011). Ela consiste na criação de um conjunto fixo de pares $[F, CR]$ que são escolhidos aleatoriamente para a variação de cada indivíduo. Este conjunto foi criado de forma a fornecer ao algoritmo estratégias distintas e complementares para a busca. Ele é formado pelos seguintes pares:

1. $[F = 1.0, CR = 0.1]$
2. $[F = 1.0, CR = 0.9]$
3. $[F = 0.8, CR = 0.2]$

Como discutido na Seção 2.1, $[F = 1.0, CR = 0.1]$ é uma estratégia interessante para atacar problemas separáveis, $[F = 1.0, CR = 0.9]$ é uma boa configuração para manter a diversidade da população, e $[F = 0.8, CR = 0.2]$ é uma estratégia que favorece a intensificação da busca. É fácil ver aqui que a geração de parâmetros para a variação de um determinado indivíduo tem custo $O(1)$.

Autoadaptação

Como dito anteriormente, neste tipo de abordagem os parâmetros são acoplados às soluções e passam a sofrer também a ação dos operadores de variação e seleção. Parâmetros acoplados a boas soluções tem uma maior chance de sobreviver. Dessa forma existe

[2]Sendo $X = \{x_1, x_2, ..., x_n\}$, $\text{lehmer}(X) = \sum_{i=1}^{n} x_i^2 / \sum_{i=1}^{n} x_i$

também uma espécie de "aprendizado", no qual os valores dos parâmetros são gradativamente refinados de acordo com a qualidade das soluções que geram. Essa abordagem está presente em (Pedrosa Silva, Lopes and Guimarães 2011) e (Teo 2006). O procedimento para a geração de novos valores de parâmetros é o próprio DE e é apresentado pela Eq.2.19:

$$\mathbf{P'}_{t,i,j} = \begin{cases} \mathbf{P}_{t,r1,j} + F'(\mathbf{P}_{t,r2,j} - \mathbf{P}_{t,r3,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR' \vee j = \delta_i \\ \mathbf{P}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{2.19}$$

$\mathbf{P}$ é um vetor de parâmetros do algoritmo. F' e CR' são parâmetros utilizados pela abordagem e devem ser estabelecidos previamente. A sensibilidade do método a estes parâmetros, no entanto, costuma e deve ser menor. Como podemos observar o procedimento descrito pela Eq.2.19 tem custo $O(1)$ por parâmetro presente em $\mathbf{P}$.

2.2.2 Configuração de Estratégias

A configuração automática de estratégias de variação já vem sendo estudada há algum tempo pela comunidade de Algoritmos Genéticos (AGs) (Thierens 2005), (Fialho, Schoenauer and Sebag 2009) e (Maturana and Saubion 2008). Para o DE não é diferente, existem na literatura inúmeras técnicas com esse objetivo, vide (Qin and Suganthan 2005), (Fialho, Ros, Schoenauer and Sebag 2010) e (Li, Fialho and Kwong 2011). Estas técnicas configuram dinamicamente operadores de mutação como os apresentados na Seção 2.1.4. A justificativa para esta adaptação dinâmica é que assim como para os parâmetros, o operador de variação ótimo pode depender do estágio em que se encontra o processo de otimização (Gong, Fialho and Cai 2010).

Todas as abordagens citadas acima são abordagens adaptativas, ou seja, abordagens que fazem uso direto de informação do processo de busca para o controle da estrutura do algoritmo. Estas técnicas podem ser dividas em duas fases:

1. Atribuição de Crédito: Define uma medida numérica de qualidade do operador e a política de atribuição;

2. Seleção de Estratégias: Define a política de seleção de operadores baseada nos créditos atribuídos.

A seguir serão apresentadas as principais abordagens relacionadas a cada uma destas fases.

Atribuição de Crédito

A medida S mais comum para avaliar a qualidade de um operador é a melhora no *fitness* trazida pelos descendentes. Essa melhora pode ser em relação:

1. Aos próprios pais (Fialho, Schoenauer and Sebag 2009);

$$S = df - pf \tag{2.20}$$

   onde pf é o valor de *fitness* do pai e df é o *fitness* do descendente gerado por ele.

2. À mediana corrente (Julstrom 1995);

$$S = df - mediana(Popf) \tag{2.21}$$

   onde $Popf$ é a lista dos valores de *fitness* de todas as soluções da população corrente.

3. Ao melhor indivíduo da população (Davis 1989);

$$S = df - bf \tag{2.22}$$

   onde bf é o *fitness* do melhor indivíduo.

Além destas medidas, uma forma de atribuição de crédito bem comum é baseada no "sucesso" do operador em criar um descendente melhor que o pai. Este tipo de crédito pode ser visto em (Qin, Huang and Suganthan 2009) e (Brest, Boskovic, Greiner, Zumer and Maucec 2007), e é descrito pela Eq.2.23.

$$S = \begin{cases} 0, & \text{se} \quad pf > df \\ 1, & \text{caso contrário} \end{cases} \tag{2.23}$$

Como visto anteriormente a diversidade da população é fundamental para o processo de busca. Por isso em (Maturana and Saubion 2008) e (Maturana, Fialho, Saubion,

Schoenauer and Sebag 2009) é proposta a medida de qualidade, descrita pela Eq.2.24, que combina melhora no *fitness* e diversidade.

$$S = \sigma * Fit + (1 - \sigma) * Div \tag{2.24}$$

onde Fit são os créditos atribuídos devido a melhora no *fitness* e Div são os créditos atribuídos devido à diversidade. σ é o coeficiente que pondera as medidas.

A diversidade pode ser calculada de diversas maneiras. Duas medidas bastante comuns para indivíduos com representação real são a variância e o somatório de todas as distâncias par a par. Para efeito de cálculo da qualidade de um operador específico a diversidade pode ser a contribuição das soluções geradas por aquele operador em relação à diversidade total da população. Considerando a uma solução gerada por um determinado operador e dist(x, y) a distância Euclidiana entre as soluções x e y, Div devido à criação de a pode ser calculado da seguinte forma.

$$Div_a = \frac{\sum_{i=1}^{NP} \text{dist}(a, i)}{\sum_{i=1}^{NP} \sum_{j=1}^{NP} \text{dist(i,j)}} \tag{2.25}$$

Como mostrado em (Wineberg and Oppacher 2003) o somatório das distâncias par a par e a variância são medidas de diversidade extremamente semelhantes. Além disso este mesmo trabalho mostra que com algumas manipulações ambas podem ser calculadas com custo $O(n)$, onde n representa o número de indivíduos.

Uma vez que a medida de qualidade é escolhida, deve-se definir a estratégia de atribuição. Usualmente, cada aplicação de um determinado operador gera um valor de crédito, estes créditos são então transformados em recompensas atribuídas a cada operador. Sendo S_k o conjunto dos créditos recebidos por cada aplicação do operador k , $k \in \{1, 2, ..., K\}$, a recompensa r_k pode ser calculada com uma das seguintes formas (Gong, Fialho and Cai 2010).

- **Recompensa Absoluta Média**

$$r_k = \frac{\sum_{i=1}^{|S_k|} S_k(i)}{|S_k|} \tag{2.26}$$

onde $|S_k|$ é a cardinalidade do conjunto S_k. Se $|S_k| = 0$, $r_k = 0$.

- **Recompensa Normalizada Média**

$$r'_k = \frac{\sum_{i=1}^{|S_k|} S_k(i)}{|S_k|}; \text{ e } r_k = \frac{r'_k}{\max\limits_{b=1,...,K} r'_b} \tag{2.27}$$

- **Recompensa Absoluta Extrema**

$$r_k = \max_{i=1,...,|S_k|} S_k(i) \tag{2.28}$$

- **Recompensa Normalizada Extrema**

$$r'_k = \max_{i=1,...,|S_k|} S_k(i); \text{ e } r_k = \frac{r'_k}{\max\limits_{b=1,...,K} r'_b} \tag{2.29}$$

Seleção de Estratégias

Distribuídas as recompensas, a segunda parte da configuração de estratégias consiste na política de seleção que determina qual estratégia será aplicada num determinado momento. Normalmente os métodos utilizados nesta fase mantêm um vetor de qualidade $Q_t = \{q_{t,1}, ..., q_{t,k}\}$ que armazena uma estimativa das recompensas r_k recebidas por cada operador ou estratégia de variação até o tempo t. Um dos métodos mais simples e que aparece com frequência na literatura relacionada ao DE é o Probability Matching descrito a seguir.

- Probability Matching (PM)

A ideia básica por trás do Probability Matching é, como o próprio nome indica, atribuir a cada operador uma probabilidade de execução p_k proporcional à sua estimativa de qualidade q_k. Para tal, considerando um conjunto de K operadores, este método mantém um vetor de probabilidades $P_t = \{p_{t,1}, ..., p_{t,k}\} (0 \leq p_{t,i} \leq 1; \sum_{i=1}^{k} p_{t,i} = 1)$ que é atualizado da seguinte forma:

$$p_{t,k} = \frac{q_{t,k}}{\sum_{i=1}^{K} q_{t,i}} \tag{2.30}$$

Definidas as probabilidades, a estratégia de variação k a ser aplicada pode então ser determinada através de um algoritmo de roleta que tem custo $O(k)$.

A aplicação deste método para a alocação de operadores no DE pode ser vista em (Gong, Fialho and Cai 2010), (Qin, Huang and Suganthan 2009) e (Qin and Suganthan 2005). Mais recentemente, em (Fialho, Ros, Schoenauer and Sebag 2010) outros dois métodos são aplicados para a configuração de estratégias do DE. Estes métodos são descritos a seguir.

- Adaptive Pursuit (AP)

No Adaptive Pursuit, assim como no Probability Matching, é mantido um vetor de probabilidades de execução P_t e novamente os operadores são selecionados por um algoritmo de roleta. A diferença entre este método e o anterior está na atualização das probabilidades $p_{t,k}$. Sendo $k^* = \arg\max_k(q_{t,k})$ o melhor operador corrente, temos:

$$p_{t+1,k*} = p_{t,k*} + \beta(P_{max} - p_{t,k*}) \tag{2.31}$$

e

$$\forall k \neq k* : p_{t+1,k} = p_{t,k} + \beta(P_{min} - p_{t,k}) \tag{2.32}$$

sob a restrição

$$P_{max} = 1 - (K-1)P_{min} \tag{2.33}$$

onde P_{min} deve ser definido pelo usuário e a restrição descrita pela Eq.2.33 garante que se $\sum_{i=1}^{K} p_{t,i} = 1$ então o somatório das probabilidades atualizadas também será igual a 1.

Enquanto o Probability Matching determina cada probabilidade proporcionalmente à estimativa da qualidade, o Adaptive Pursuit implementa um esquema do estilo "o vencedor leva tudo", que incrementa rapidamente a probabilidade da melhor estratégia corrente.

- Multi-Armed Bandit (MAB)

Diferente das abordagens apresentadas anteriormente, no MAB a decisão sobre qual estratégia executar é tomada deterministicamente. Formalmente, considere $n_{t,k}$ como o

número de vezes que o operador k foi executado até o tempo t, e $p_{t,k}$ a média das recompensas recebidas por k. O método então seleciona no instante de tempo t o operador que maximiza a seguinte quantidade:

$$p_{t,j} + C\sqrt{\frac{\log \sum_k n_{t,k}}{n_{t,j}}} \tag{2.34}$$

O termo da esquerda da Eq.2.34 favorece a opção com a maior recompensa média, já o termo da direita favorece a exploração dentre os operadores disponíveis. C é um parâmetro do método que controla a relação de compromisso entre os dois termos.

2.2.3 Seleção de Estratégias com Q-learning

Nesta seção é apresentada a metodologia proposta para a seleção de estratégias de variação do DE, utilizando o método de aprendizado por reforço conhecido como Q-learning.

Aprendizado por reforço é o problema enfrentado por um agente que precisa aprender um comportamento através de interações do tipo tentativa e erro com um ambiente dinâmico. No modelo padrão de aprendizado por reforço (Kaelbling, Littman and Moore 1996), o agente é conectado ao ambiente de duas formas: percepção e ação. Ao interagir com o ambiente, o agente recebe informação sobre o estado atual e escolhe uma ação a ser executada. Esta ação causa uma alteração no ambiente que retorna ao agente um sinal de reforço que pode ser uma recompensa ou uma penalidade.

Como pode-se observar, o problema de configuração de operadores se encaixa perfeitamente à estrutura de aprendizado por reforço. O DE pode ser visto como um agente que aprende através de interações com o ambiente, no caso a população de indivíduos. O DE interage com essa população através dos operadores de variação e o sinal de reforço é calculado através da avaliação da população gerada. Com isso, o DE pode então aprender a melhor política de utilização dos operadores.

O Q-learning (Watkins 1989) é uma forma de aprendizado por reforço livre de modelo. É livre de modelo no sentido de que fornece aos agentes a capacidade de aprender como agir, simplesmente, pela experimentação das consequências de suas ações. O aprendizado é todo feito sem a necessidade de construção de um mapeamento do domínio, e acontece

da seguinte forma (Watkins and Dayan 1992): o agente executa uma ação a, num determinado estado s, avalia sua consequência em termos da recompensa recebida r e da estimativa do valor do estado para o qual ele foi, como descrito pela Eq.2.35.

$$Q(s_t, a_t) \leftarrow \lambda(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})) + (1 - \lambda)Q(s_t, a_t) \tag{2.35}$$

onde Q é a recompensa esperada pela execução da ação a no estado s, $\max_{a'} Q(s', a')$ é uma estimativa da melhor recompensa futura a ser recebida a partir do estado s', e γ é um fator de desconto aplicado a esta estimativa.

Ao tentar todas as ações em todos os estados repetidamente o agente aprende a melhor política julgando pela recompensa de longo prazo e não apenas pela recompensa imediata. Na prática, no início do processo o agente explora o espaço de ações quase que aleatoriamente, então a taxa de exploração é continuamente reduzida de forma que o agente passa a escolher as ações que ele "pensa" serem as melhores naquele estado.

Em (Guo, Liu and Malec 2004) é proposta uma forma de controlar a exploração do método baseada no Critério de Metrópolis oriundo da técnica de Arrefecimento Simulado (Kirkpatrick, Gelatt and Vecchi 1983). Nela a diferença entre recompensas estimadas para a melhor ação corrente e para uma ação aleatória é levada em consideração para decidir se o método irá fazer uma exploração do espaço de ações ou não. Este processo é ainda controlado por uma variável de "temperatura" que garante uma alta exploração no início e uma intensificação no final. No trabalho citado, este procedimento apresentou um desempenho superior quando comparado com outras formas de controle da exploração do Q-learning. Por este motivo ele foi incorporado no método proposto como podemos observar no Algoritmo 2, linha 7.

Para a configuração de estratégias de variação consideraremos apenas um estado, no qual qualquer um dos operadores pode ser sempre executado. Assim, utilizaremos uma adaptação do *Q-learning de Um Estado* (Wunder, Littman and Babes 2010).

Para cada indivíduo o DE escolhe uma estratégia de variação a ser aplicada. Para cada vetor teste $\mathbf{u}_{t,i}$ criado, uma recompensa, baseada em alguma medida de qualidade é calculada. Então, o valor Q para a estratégia ou ação escolhida a_t é calculado (Eq.2.36). Através deste processo o DE aprende quais são as estratégia que maximizam sua recompensa. Se a recompensa for baseada em melhora no *fitness*, o DE aprenderá qual estratégia produz consecutivamente boas soluções. Caso a recompensa seja baseada em na diversidade, o DE aprenderá qual estratégia produz soluções que mais contribuem

para a diversidade da população. O Algoritmo 2 descreve o método proposto.

$$Q(a_t) \leftarrow \lambda(r + \gamma \max_{a_{t+1}} Q(a_{t+1})) + (1 - \lambda)Q(a_t) \tag{2.36}$$

Algoritmo 2: Algoritmo Q-learning com Critério de Metrópolis

1 **ENTRADAS:**
2 $\lambda \leftarrow$ taxa de aprendizado , $\gamma \leftarrow$ fator desconto, $temperatura \leftarrow 1$;
3 **INÍCIO**
4 $\forall_a$ Q(a) $\leftarrow 0$;
5 $ar_t \leftarrow$ ação aleatória;
6 $ab_t \leftarrow$ arg max$_\alpha$Q(a_t);
7 **if** *rand()* $<$ $exp((Q(ar_t) - Q(ab_t))/temperatura)$ **then**
8 $\quad a_t \leftarrow ar_t$;
9 **else**
10 $\quad a_t \leftarrow ab_t$;
11 **end**
12 Executa a ação a_t;
13 Recebe a recompensa r;
14 Q(a_t) $\leftarrow \lambda(r + \gamma$ max$_{a_{t+1}}$Q(a_{t+1})) + (1 - λ)Q(a_t);
15 $\lambda \leftarrow 0.99^*\lambda$;
16 $temperatura \leftarrow 0.98^*temperatura$;
17 **goto** 6;
18 **FIM**

Quando o algoritmo precisa determinar a melhor ação corrente pelo procedimento, $ab_t \leftarrow \arg\max_\alpha Q(a_t)$ (Algoritmo 2 linha 7) e ocorre um empate, a decisão é feita de forma aleatória entre os empatados.

A utilização do Q-learning muda consideravelmente a lógica de bonificação das estratégias uma vez que, além da recompensa imediata, também considera a estimativa de recompensa futura. Assim podemos dizer, que de certa forma, o mecanismo de atribuição de crédito no Q-learning é menos imediatista.

2.3 Algoritmo de Evolução Diferencial com Configuração Automática: Estrutura Básica

Até aqui foram descritos o DE básico, abordagens para a configuração de parâmetros e abordagens para a configuração de estratégias de variação. Nesta seção será descrita uma estrutura básica que agrupa todos estes mecanismos.

Existem várias formas de fazer com que o DE se configure automaticamente. A forma mais comum é fornecer a ele um conjunto de parâmetros e um conjunto de estratégias que podem ser adaptados através das técnicas descritas até aqui. Na literatura (Ver (Qin, Huang and Suganthan 2009), (Qin and Suganthan 2005), (Brest, Boskovic, Greiner, Zumer and Maucec 2007), (Brest, Greiner, Boskovic, Mernik and Zumer 2006)), os parâmetros costumam estar acoplados às soluções. Neste caso, os operadores de variação, que agem sobre uma determinada solução, são configurados pelos parâmetros acoplados à ela. O problema desta abordagem é que soluções boas costumam ser mais difíceis de serem melhoradas. Assim, bons parâmetros que estejam acoplados a boas soluções terão baixa chance de se perpetuar pois terão dificuldade em gerar descendentes melhores para aquela solução.

Para contornar esta desvantagem, no algoritmo proposto, desacoplam-se os parâmetros das soluções. Dessa forma, além da população de indivíduos X_t com NP indivíduos, utiliza-se uma população P_t de parâmetros de mesma cardinalidade. Um indivíduo da população de parâmetros é definido da seguinte forma:

$$p_{t,i} = \{F_{t,i,1}, CR_{t,i,1}, F_{t,i,2}, CR_{t,i,2}, ..., F_{t,i,k}, CR_{t,i,k}\} \tag{2.37}$$

onde k é o índice da estratégia de variação. Ou seja, para cada estratégia de variação diferente, temos um par de parâmetros correspondente. É importante mencionar que para as estratégias sem recombinação (ver Eq.2.12) $CR_{t,k}$ corresponde ao parâmetro Z.

O conjunto de estratégias de variação é composto por `rand/1`, `rand/2` ,`current-to-best/2` e `current-to-rand/2`. Devido a sua diversidade, esse é um conjunto bastante comum em versões do DE com configuração automática de estratégias, vide (Qin and Suganthan 2005), (Qin, Huang and Suganthan 2009), (Fialho, Ros, Schoenauer and Sebag 2010) e (Wang, Cai and Zhang 2011). Para a seleção de estratégias é mantido o vetor de qualidades $Q_t = \{q_{t,1}, q_{t,2}, ..., q_{t,k}\}$.

Para atualizar a tabela de qualidades foi adotado o procedimento criado por Fialho, Costa, Schoenauer and Sebag (2008), no qual existe uma janela W_k, de tamanho $l = 10$, que contêm as últimas recompensas recebidas pela estratégia k. A recompensa que a estratégia receberá de fato é a melhor da janela. Este método foi proposto devido aos resultados apresentados por Whitacre, Pham and Sarker (2006) que mostram que é melhor selecionar estratégias que produzam soluções *outliers* do que aquelas que produzem boas soluções em média. Pelo mesmo motivo o método de cálculo de recompensas empregado aqui será o de Recompensa Normalizada Extrema descrito pela Eq.2.29. A normalização é importante, pois a magnitude das melhoras no *fitness* muda muito durante o processo de otimização.

Em síntese, para cada indivíduo da população de soluções, primeiramente seleciona-se uma estratégia de variação através da tabela de qualidades. Após, seleciona-se aleatoriamente, sem repetição, um indivíduo da população de parâmetros. Este indivíduo sofre a ação de algum dos operadores de configuração de parâmetros apresentados na Seção 2.2.1. Note que esta ação é feita somente nos parâmetros referentes à estratégia selecionada. Com os novos parâmetros a solução é variada pela estratégia selecionada. Por fim, se o vetor teste $\mathbf{u}$, for melhor que o vetor corrente $\mathbf{x}$ (ver Eq. 2.4), então o vetor teste e os parâmetros modificado são selecionados. Caso contrário, o vetor corrente é selecionado e as modificações realizadas nos parâmetros são descartadas. O Algoritmo 3 descreve todo o processo.

2.4 Considerações Finais

No presente capítulo foi realizada uma explanação profunda a cerca do algoritmo de Evolução Diferencial, na qual discutiu-se o funcionamento dos seus principais operadores, bem como o papel e a influência dos parâmetros de controle destes operadores. Foi feita também uma revisão extensa sobre métodos de controle automático de parâmetros e de estratégias para o DE, e uma nova abordagem para a seleção de operadores de variação baseada em aprendizado por reforço foi proposta.

Conforme apresentado nos tópicos anteriores, os métodos estudados apresentam diferentes características funcionais, dentre as quais pode-se citar, por exemplo, o tipo de mecanismo de adaptação empregado e a presença de aprendizado ou não. A fim de se verificar experimentalmente o efeito e a relevância destes fatores para o desempenho do métodos descreveu-se então um arcabouço genérico que comporta as diferentes

Algoritmo 3: Algoritmo de Evolução Diferencial com Configuração Automática

$t \leftarrow 1$
Inicializar População $X_t\{\mathbf{x}_{t,i}; i = 1, 2, ..., NP\}$
Inicializar População de Parâmetros $P_t = \{\mathbf{p}_{t,i} = \mathcal{U}_{[0,1]}; i = 1, 2, ..., NP\}$
Inicializar Tabela de Qualidades $Q_t = \{\mathbf{q}_{t,i} = 0; i = 1, 2, ..., k\}$
Inicializar Janelas de Recompensas $W_{t,k} = \{\mathbf{r}_{t,i,j} = 0; i = 1, 2, ..., k, j = 1, 2, ..., S\}$
Avalia X_t
while not condição_de_parada **do**
  for $i = 1$ to NP **do**
    Selecione uma Estratégia de Variação e
    Escolher um índice aleatório sem reposição $j \in NP$
    $\mathbf{p}'_{t,j,e}$ = Configura ($\mathbf{p}_{t,j,e}$)
    Produzir o vetor teste $\mathbf{u}_{t,i}$ utilizando a estratégia e e parâmetros $\mathbf{p}'_{t,j,e}$
    Avaliar $\mathbf{u}_{t,i}$
    Calcular o crédito S devido à criação da solução $\mathbf{u}_{t,i}$
    Fazer seleção
    if $\mathbf{u}_{t,i}$ for selecionado **then**
      $\mathbf{p}_{t,j,e} = \mathbf{p}'_{t,j,e}$
      Calcular o crédito S devido à criação da solução $\mathbf{u}_{t,i}$
    end if
  end for
  $\forall k$ Calcular Recompensas r_k
  $\forall k$ Atualizar$W_{t,k}$
  AtualizarQ_t
  $t \leftarrow t + 1$
end while

abordagens. O capítulo a seguir descreve as configurações testadas e o planejamento experimental utilizado para se averiguar tais efeitos.

Capítulo 3

Experimentos Computacionais

3.1 Configuração dos Experimentos

Na Seção 2.2.1 do Capítulo 2 foram apresentados 4 grupos de estratégias de configuração automática de parâmetros. Como pudemos observar, com exceção da abordagem baseada em conjunto fixo de estratégias, todas as outras apresentam seu próprio conjunto de parâmetros. Apesar do DE ser menos sensível a estes parâmetros, sabe-se também que eles não deixam de exercer influência no desempenho do método. Assim, nos grupos em que existe mais de uma abordagem foi utilizado como critério de escolha a sua frequência na literatura e a estabilidade dos seus parâmetros dentre os trabalhos relacionados. Desta forma, temos as seguintes possibilidades:

- Aleatorização com distribuição uniforme (**u**):

$$P'_{g,i} = \mathcal{U}_{[a,b]} \text{ , se } \mathcal{U}_{[0,1]} \leq \tau \tag{3.1}$$

  - $a = \begin{cases} 0 & \text{se} \quad P' = CR \\ 0.1 & \text{se} \quad P' = F \end{cases}$
  - $b = 1$
  - $\tau = 0.1$

- Aleatorização com Adaptação (**aa**): Em acordo com (Qin, Huang and Suganthan 2009) e (Qin and Suganthan 2005) este procedimento é aplicado somente ao parâmetro

CR, e para o parâmetro F é feita uma aleatorização simples com distribuição normal.

$$\begin{aligned} CR'_{g,i} &= \mathcal{N}_{[P_m,a]} \\ F'_{g,i} &= \mathcal{N}_{[b,c]} \end{aligned} \tag{3.2}$$

- $P_m = \text{mediana}(L_P)$
- $L_g = 20$ gerações
- $a = 0.1$
- $b = 0.5$
- $c = 0.3$

- Conjunto fixo de estratégias (**cf**):

$$[F', CR'] = \begin{cases} [1.0, 0.1] & \text{se} \quad \mathcal{U}_{[0,0.9]} \leq 0.3 \\ [1.0, 0.9] & \text{se} \quad 0.3 < \mathcal{U}_{[0,0.9]} \leq 0.6 \\ [0.8, 0.2] & \text{se} \quad 0.6 < \mathcal{U}_{[0,0.9]} \leq 0.9 \end{cases} \tag{3.3}$$

- Autoadaptação (**de**):

$$\mathbf{P'}_{t,i,j} = \begin{cases} \mathbf{P}_{t,r1,j} + F'(\mathbf{P}_{t,r2,j} - \mathbf{P}_{t,r3,j}), & \text{se } \mathcal{U}_{[0,1]} \leq CR' \vee j = \delta_i \\ \mathbf{P}_{t,i,j}, & \text{caso contrário} \end{cases} \tag{3.4}$$

- $F' = \mathcal{U}_{[0.6,1]}$
- $CR' = \mathcal{U}_{[0.9,1]}$

Além destas, como controle foram testadas as três configurações fixas (configurações presentes em **cf**), e uma configuração com aleatorização que utiliza o intervalos para F e CR recomendados em (Price, Storn and Lampinen 2005). São elas:

- **px**:

$$[F = 1.0, CR = 0.1] \tag{3.5}$$

- **py**:

$$[F = 1.0, CR = 0.9] \tag{3.6}$$

- **pz**:

$$[F = 0.8, CR = 0.2] \tag{3.7}$$

- **ps**:

$$[F = \mathcal{U}_{[0.4,1]}, CR = \mathcal{U}_{[0.9,1]}] \tag{3.8}$$

Em relação à configuração de estratégias de variação foram realizados testes com os seguinte métodos:

- Q-learning (**q**)
- *Pobability Matching* (**p**)

O método Q-learning ainda possui o fator de desconto γ para a estimativa de recompensa futura. Este parâmetro, por sua vez, será testado com 4 níveis $\{0.1, 0.3, 0.7, 1\}$. Cada um dos métodos de seleção de estratégia poderá trabalhar, ainda, com um dos dois tipos de atribuição de crédito:

- Atribuição de Crédito:
  - Baseada na melhora do *Fitness* em relação aos pais (**f**) (Eq. 2.20)
  - Baseada em Diversidade (**d**) (Eq. 2.25)

Em ambos os casos, os operadores de variação só recebem os créditos se produzirem soluções melhores que os pais, caso contrário o crédito atribuído tem valor 0.

Dentre os métodos apresentados na seção 2.2.2 o *Pobability Matching* foi escolhido para comparação, pois ele é livre de parâmetros, é o mais comum na literatura e seus resultados são competitivos com qualquer uma das outras técnicas citadas, como pode ser visto em (Fialho, Ros, Schoenauer and Sebag 2010). Além dele, como controle foram feitos testes utilizando apenas uma das estratégias do conjunto e uma versão que escolhe aleatoriamente alguma das 4 estratégias. Estas configurações são descritas a seguir:

- (**1**): `rand/1` (Eq.2.10)
- (**2**): `current-to-best/2` (Eq.2.9)
- (**3**): `rand/2` (Eq.2.11)
- (**4**): `current-to-rand/1` (Eq.2.12)
- Aleatório (**u**):

$$\begin{cases} (1) & \text{se} \quad \mathcal{U}_{[0,1]} \leq 0.25 \\ (2) & \text{se} \quad 0.25 < \mathcal{U}_{[0,1]} \leq 0.5 \\ (3) & \text{se} \quad 0.5 < \mathcal{U}_{[0,1]} \leq 0.75 \\ (4) & \text{se} \quad 0.75 < \mathcal{U}_{[0,1]} \end{cases} \tag{3.9}$$

3.2 Metodologia

Definidas as configurações de cada método, os experimentos terão a seguinte forma:

- Três fatores serão analisados:
  - Tipo de configuração de parâmetros (TCP) (8 níveis)
    - Aleatório (**u**)
    - Aleatório com Adaptação (**aa**)
    - Conjunto Fixo de Estratégias (**cf**)
    - Autoadaptação (**de**)
    - Configuração fixa 1 (**px**)
    - Configuração fixa 2 (**py**)
    - Configuração fixa 3 (**pz**)
    - Configuração Recomendada (**ps**)
  - Tipo de configuração de estratégias de variação (TCE) (10 níveis)
    - Com aprendizado (TCEA)
      - Q-Learning ($\gamma = 0.1$) (**q0.1**)
      - Q-Learning ($\gamma = 0.3$) (**q0.3**)
      - Q-Learning ($\gamma = 0.7$) (**q0.7**)

        - · Q-Learning ($\gamma = 1$) (**q1**)
        - · *Pobability Matching* (**p**)
    - ∗ Sem aprendizado (TCES)
        - · `rand/1` (**1**)
        - · `current-to-best/2` (**2**)
        - · `rand/2` (**3**)
        - · `current-to-rand/1` (**4**)
        - · Aleatório (**u**)
- Tipo de recompensa (TREC) (2 níveis)
    - ∗ Baseada em *fitness* (**f**)
    - ∗ Baseada em diversidade (**d**)

Lembrando que somente as abordagens de seleção com aprendizado fazem uso de recompensa, teremos $TCP * (TCEA * TREC + TCES)$ configurações possíveis. Ou seja, 120 variações do Algoritmo 3. Para a avaliação dos fatores descritos, foi utilizado o Black Box Optimization Benchmark (BBOB 2012)[1]. Este *benchmark* possui um conjunto de 24 funções[2] não lineares mono-objetivo irrestritas com variáveis reais. Estas funções são divididas em 5 grupos:

1. Separáveis (5 funções);
2. Funções com condicionamento moderado ou baixo (4 funções);
3. Funções unimodais com com alto condicionamento (5 funções);
4. Multimodais com estrutura global adequada (5 funções); e
5. Multimodais com estrutura global fraca (5 funções).

Os valores ótimos de todos os problemas são conhecidos e cada um pode variar de 2 a 40 dimensões.

Devido a natureza estocástica dos métodos, cada variação foi executada 15 vezes em cada um dos 24 problemas com 5, 10 e 20 dimensões. Como condição de parada dos

[1]Disponível em http://coco.gforge.inria.fr/doku.php?id=bbob-2012

[2]Para uma completa descrição dos problemas ver (Finck, Hanseny, Raymond and Augerx 2010), disponível em http://coco.lri.fr/BBOB-downloads/download11.05/bbobdocfunctions.pdf

métodos, foram utilizados 2 critérios. (i) número de avaliações de função maior do que 10^6 e (2) $f_b - f_{opt} < 10^{-8}$. Onde f_b é o valor de função da melhor solução encontrada pelo método e f_{opt} o valor de função no ponto ótimo. Para cada execução foram coletadas duas medidas:

- Número de avaliações de função para convergência[3] (Nfeval)
- Diferença em relação ao ótimo ($f_b - f_{opt}$)

Para a análise estatística dos resultados, como houve violação das premissas da análise de variância (ANOVA) comum, foi utilizado um modelo modificado. Este modelo, descrito em (Anderson and Robinson 2001), trabalha com testes de permutação no lugar de testes baseados na teoria de normalidade.

Para todas as análises foi considerado um grau de significância $\alpha = 0.05$. Para os casos em que constatou-se efeito significativo ($p < 0.05$), foram feitas análises post-hoc através de testes de comparação múltipla de médias com contraste sequencial. Nelas, os métodos são ordenados pela média obtida para cada variável de resposta. O teste de hipótese verifica se a diferença entre o resultado de um método e de seu consecutivo é 0.

Como a computação deste modelo é custosa e o objetivo da análise é comparar o desempenho médio dos métodos, os resultados das 15 execuções em cada problema foram colapsados na média[4]. Este procedimento é justificado estatisticamente em (Montgomery 2008).

3.3 Resultados

A análise de variância mostrou que:

- Não houve efeito significativo do tipo de recompensa, para nenhuma das variáveis de resposta;
- Houve efeito significativo do tipo de configuração de parâmetros (TCP) para o número de avaliações de função;

[3]Considera-se que o método convergiu se $f_b - f_{opt} < 10^{-8}$. Caso o método não convirja antes que algum critério de parada seja atendido, Nfeval é igual a 10^6.

[4]Os resultados completos estão disponíveis online em: https://sites.google.com/site/rcpsilva/results-and-codes

- Houve efeito significativo do tipo de configuração de estratégias (TCE) para o número de avaliações de função e para a diferença em relação ao ótimo;
- Houve interação significativa entre TCP e TCE para ambas as variáveis de resposta; e
- Não houve outras interações significativas.

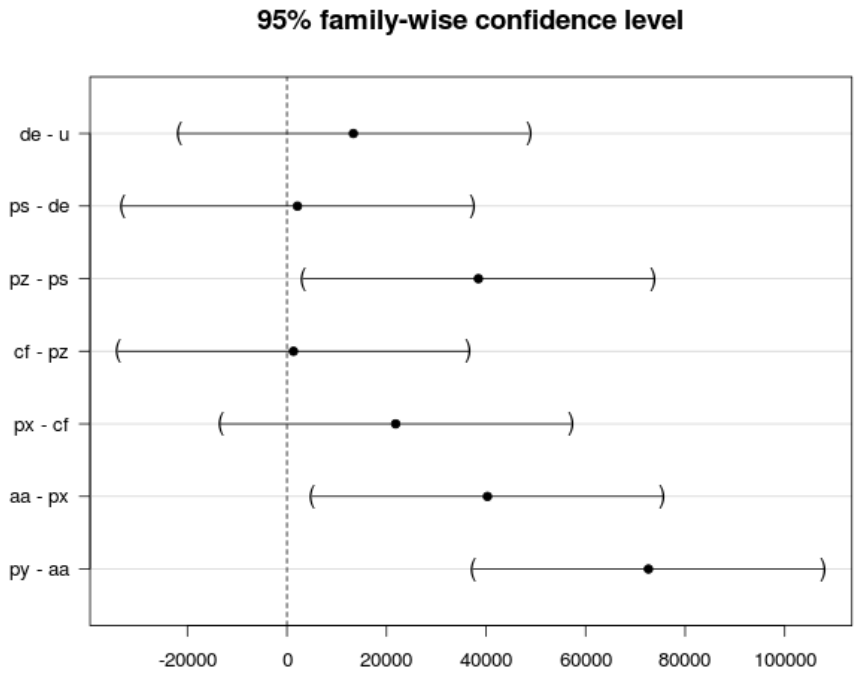
Ambas as recompensas são atribuídas a operadores que produziram soluções melhores que os pais e parece que só a alteração da medida de qualidade não foi suficiente para imprimir diferença entre os métodos.

As figuras a seguir mostram as diferenças entre as médias e os respectivos intervalos de confiança. Se o intervalo contém o 0, significa que não há diferença significativa entre os métodos.

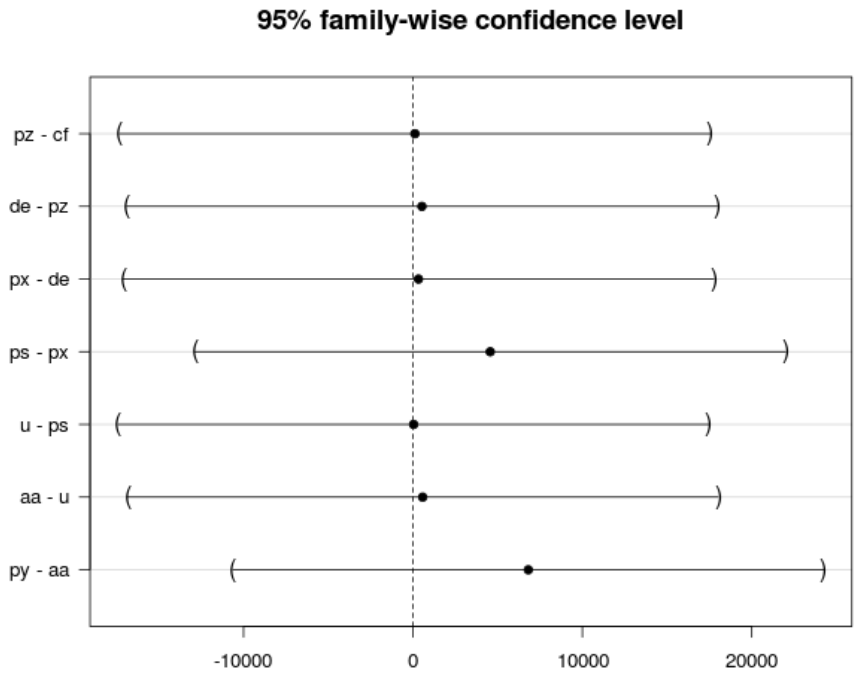
A Figura 3.1 mostra os resultados do tipo de configuração de parâmetros. Em relação ao número de avaliações de função, Figura 3.1-(a), temos a presença de 3 blocos. O primeiro bloco, que apresentou os melhores resultados, é formado pelas abordagens, aleatória (**u**), autoadaptação (**de**) e aleatória com intervalos recomendados (**ps**). O segundo bloco é formado pelas estratégias fixas **px**, **pz** e pela estratégia que usa o conjunto de estratégias fixas (**cf**). Com os piores, temos as abordagens aleatória adaptada (**aa**) e a estratégia fixa (**py**).

Em relação à diferença para o ótimo, Figura 3.1-(b), como já verificado na análise de variância não houve diferença significativa entre nenhum dos métodos.

Por esta análise, é clara a divisão entre os métodos que fornecem maior diversidade de parâmetros e aqueles que trabalham com menor diversidade. Os resultados indicam que qualquer que seja o mecanismo de variação dos parâmetros, parece ser mais importante que eles mantenham certa diversidade. O resultados obtidos por **py** podem ser explicados pelo fato de esta ser uma estratégia exploratória (Wang, Cai and Zhang 2011), logo, espera-se mesmo que ela faça um número de avaliações de função maior para convergir do que as outras estratégias fixas. No mecanismo **aa**, ao fim do período de armazenamento de parâmetros de sucesso, toda a aleatorização é feita com distribuição gaussiana sobre o mesmo valor, o que limita a diversidade. Além disso, é possível que este mecanismo tenha uma tendência a produzir estratégias exploratórias, uma vez que apenas a primeira população de parâmetros é totalmente aleatória. Estes dois fatores explicariam o "atraso" do mecanismo **aa** para convergir.



(a) Número de Avaliações de Função



(b) Diferença em relação ao Ótimo

Figura 3.1: Análise post-hoc dos tipos de configuração de parâmetros

A Figura 3.2 mostra os resultados para o TCE. Em relação ao número de avaliações de função, Figura 3.2-(a),podemos notar que, os métodos com múltiplas estratégias de variação não apresentaram diferença estatística entre si, porém, apresentaram desempenho significativamente superior aos métodos com estratégica única. Sobre a diferença em relação ao ótimo, Figura 3.2-(b),apenas o método de estratégia fixa 4 apresentou desempenho significativamente inferior.

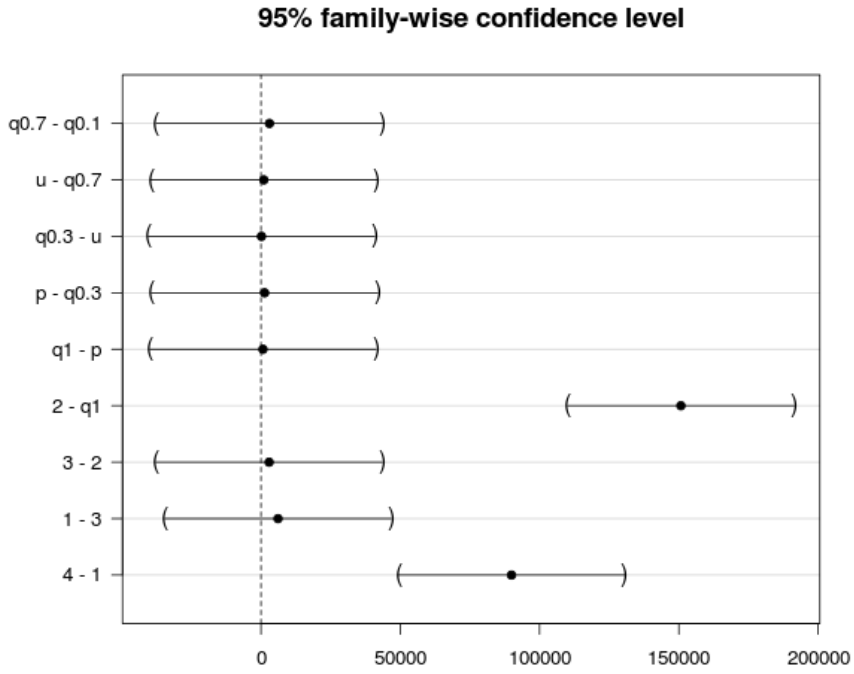
Com a exceção da abordagem **4**, todas as outras abordagens, que utilizam estratégia de variação única, apresentaram desempenho estatisticamente igual. Isso explicaria a não existência de diferença significativa dentre a abordagens com múltiplas estratégias. Uma vez que as opções são tão parecidas, não faz diferença entre escolher uma ou outra. Assim, qualquer método de aprendizado tende a ficar igual ao método de escolha aleatória.

A Figura 3.3 mostra as interações entre TCE e TCP em relação ao número de avaliações de função, (a) mostra as interações separadamente com intervalos de confiança, enquanto (b) mostra as interações sobrepostas. As diferentes abordagens de configuração de parâmetros possuem desempenho variável quando utilizam apenas um operador de variação. Por exemplo: **u**, **ps**, **de**, **py** e **aa** possuem desempenho significativamente inferior quando utilizam a estratégia **4**. **pz** e **cf** apresentam seus piores desempenhos com a estratégia **2**, já **px** apresenta desempenho inferior quando utiliza a estratégia **3**. A utilização de múltiplas estratégias, independente do tipo de seleção, elimina esta variação de desempenho, como é o caso de **aa**, **py**, **u** e **ps**. Para **px**, **pz**, **cf** e **de**, o desempenho melhora significativamente quando são utilizadas múltiplas estratégias.

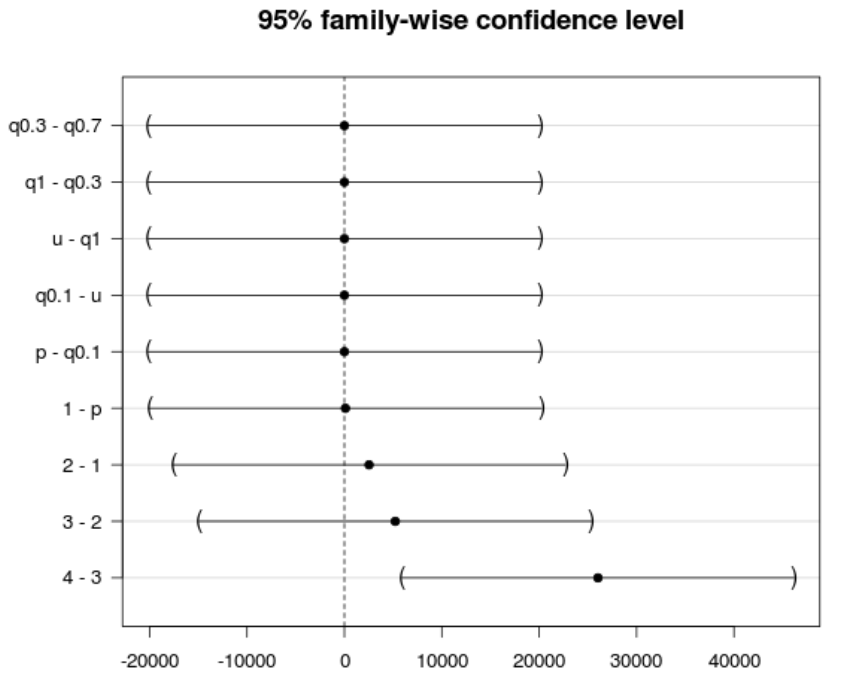
É importante notar aqui a semelhança entre os padrões de interações dos TCPs **py** e **aa**. Esta semelhança corrobora a hipótese de que o mecanismo presente em **aa** favorece estratégias mais exploratórias.

A Figura 3.4 exibe as interações entre TCE e TCP em relação à diferença para o ótimo: (a) mostra as interações separadamente com intervalos de confiança, enquanto (b) mostras as interações sobrepostas. **py** apresentou desempenho variável para estratégias fixas. **u**, **ps** e **aa** apresentaram desempenho significativamente inferior, quando utilizaram a abordagem 4. Os demais métodos mostraram desempenho semelhante para todas as configurações.

A utilização de múltiplas estratégias de variação aumenta as possibilidades de exploração do espaço de busca. Esse efeito por si só, parece abrandar as limitações impostas por uma configuração de parâmetros mais rígida. Em alguns casos, como **px** e

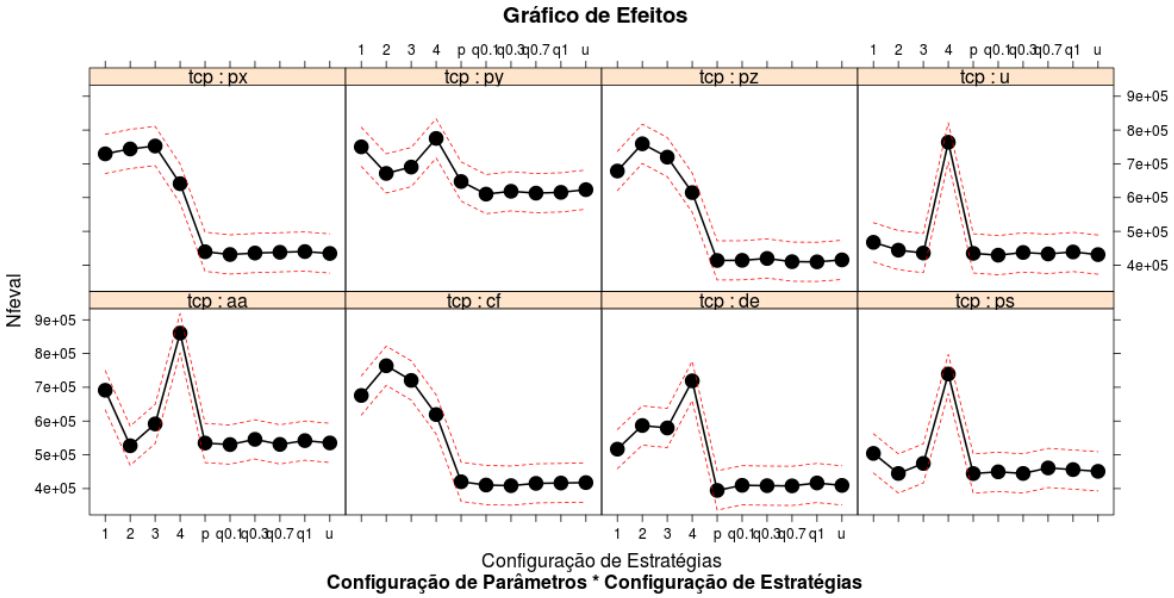


(a) Número de Avaliações de Função

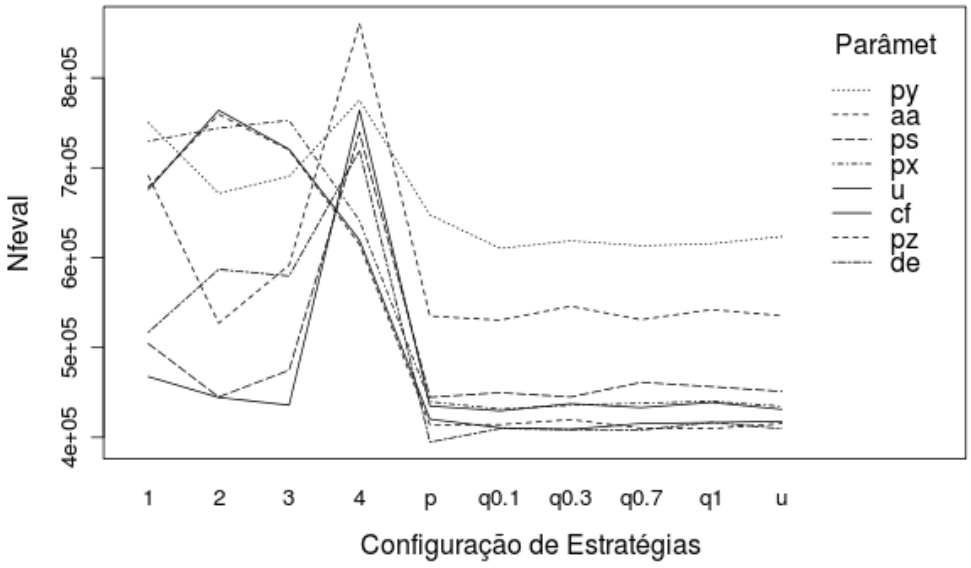


(b) Diferença em relação ao Ótimo

Figura 3.2: Análise post-hoc dos tipos de configuração de estratégias de variação

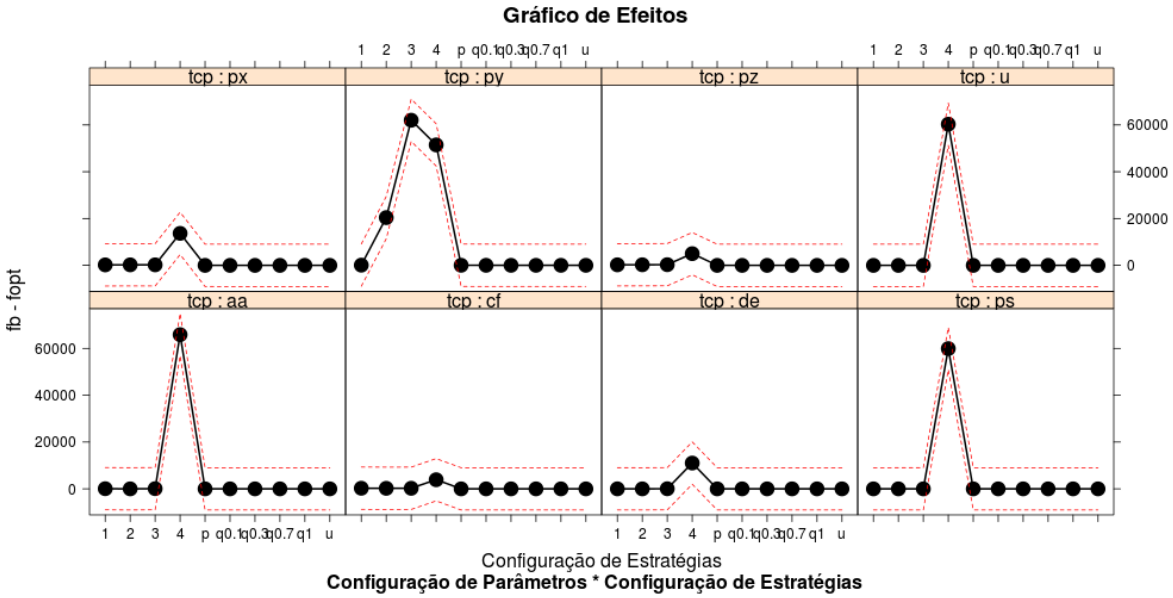


(a)

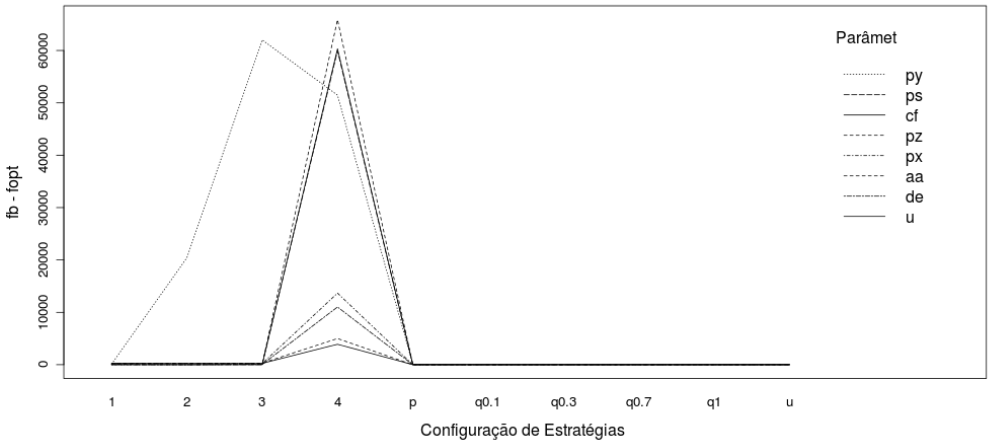


(b)

Figura 3.3: Interações (Nfeval)



(a)



(b)

Figura 3.4: Interações ($f_b - f_{opt}$)

cf, mesmo utilizando uma configuração de parâmetros bem rígida, com a utilização de múltiplas estratégias de variação, obtêm-se resultados comparáveis aos melhores obtidos por qualquer outra configuração.

Capítulo 4

Conclusão

Algoritmos Evolutivos têm sido usados desde os anos 1980 como uma ferramenta importante para a resolução de problemas difíceis de otimização. Em particular, o algoritmo de Evolução Diferencial, proposto em meados da década de 1990, vem se destacando, tanto nas variedades de problemas em que é aplicado, quanto no desempenho obtido em competições especializadas.

O DE, assim como outros algoritmos evolutivos, já demonstrou sua habilidade em tratar problemas de otimização fora do alcance de métodos tradicionais. Apesar do sucesso, seu desempenho está diretamente relacionado à escolha adequada dos operadores de variação e dos parâmetros que controlam o funcionamento destes operadores.

Para se configurar corretamente o DE, precisaria-se de certas informações sobre as características do problema em mãos. Na maioria das vezes estas informações não estão disponíveis. Assim, resta ao usuário utilizar uma configuração padrão, que pode levar à um desempenho pobre, ou realizar uma série de testes, na qual é gasta uma grande quantidade de tempo e recurso computacional. Neste contexto, têm-se buscado cada vez mais por métodos de configuração automática. Nestes métodos, o próprio algoritmo é responsável por encontrar a configuração de operadores e/ou os valores de parâmetros para o problema sendo tratado.

Existem na literatura abordagens que fazem configuração automática de parâmetros, abordagens que fazem configuração automática de operadores de variação e outras abordagens que fazem a duas coisas. Com o objetivo de entender o impacto de cada um desses fatores e de suas possíveis interações no desempenho do DE, neste trabalho, foi apresentado um algoritmo autoconfigurável genérico. Ele é capaz de trabalhar com qual-

quer combinação destas abordagens e portanto fornece uma base para que elas possam ser avaliadas experimentalmente.

Além disso, devido à semelhança do problema de alocação de operadores de variação com o problema de aprendizado por reforço, foi proposto neste trabalho uma nova abordagem baseada no algoritmo Q-learning para este fim.

Os resultados mostraram que existe um forte impacto tanto do tipo de configuração de parâmetros (TCP) quanto no tipo de configuração de estratégias (TCE) no desempenho do algoritmo.

Em relação ao TCP, métodos que garantem uma maior diversidade de parâmetros apresentaram desempenho significativamente superior. Assim, uma alternativa bastante razoável, considerando-se o custo computacional embutido, é a simples aleatorização dos parâmetros com distribuição uniforme.

Para o TCE, no geral, métodos com estratégias múltiplas apresentaram desempenho superior quando comparados aos métodos de estratégia única. Dentre os métodos com múltiplas estratégias não houve diferença significativa. Este resultado pode ser explicado pelo fato de que a diferença de desempenho da maioria das estratégias que poderiam ser escolhidas era pequena. Assim, com este conjunto de estratégias, a simples escolha aleatória também é uma opção bem razoável.

Outro resultado importante obtido é que há uma forte interação entre o TCE e o TCP. Sobre este aspecto constatou-se que a utilização de abordagens com múltiplas estratégias é sempre benéfica. Não houve abordagem de estratégia única que sempre as superasse independente do TCP.

É interessante salientar que todas as metodologias descritas neste trabalho podem ainda ser aplicadas em outros contextos. Por exemplo, configurar automaticamente outros AEs, como os algoritmos genéticos. Além disso, criando-se um método adequado para o cálculo de recompensas, os métodos de configuração de estratégias ainda podem ser usados para a seleção de estruturas de vizinhança para problemas discretos e até seleção de diferentes heurísticas.

A conclusão geral deste trabalho é que o DE original realmente tem a ganhar com a utilização de procedimentos de configuração automática. Além disso, estes procedimentos não precisam ser nem um pouco complicados. Ao menos para as estratégias de variação consideradas aqui, sua simples aleatorização e a simples aleatorização dos parâmetros de controle já foi o suficiente para melhorar as características de robustez

do DE.

Referências Bibliográficas

Anderson, M. J. and Robinson, J.: 2001, Permutation tests for linear models, *Australian & New Zealand Journal of Statistics* **43**(1), 75–88.

Arabas, J., Maitre, O. and Collet, P.: 2012, Parade: a massively parallel differential evolution template for easea, *Proceedings of the 2012 international conference on Swarm and Evolutionary Computation*, SIDE'12, Springer-Verlag, Berlin, Heidelberg, pp. 12–20.

Babu, B. V. and Jehan, M. M. L.: 2003, Differential evolution for multi-objective optimization, *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, Vol. 4, pp. 2696–2703 Vol.4.

Back, T., Hammel, U. and Schwefel, H.-P.: 1997, Evolutionary computation: Comments on the history and current state, *IEEE Transaction on Evolutionary Computation* **1**, 3–17.

Birattari, M., Stützle, T., Paquete, L. and Varrentrapp, K.: 2002, A racing algorithm for configuring metaheuristics, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 11–18.

Brest, J., Boskovic, B., Greiner, S., Zumer, V. and Maucec, M. S.: 2007, Performance comparison of self-adaptive and adaptive differential evolution algorithms, *Soft Computing* **11**, 617–629.

Brest, J., Greiner, S., Boskovic, B., Mernik, M. and Zumer, V.: 2006, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *Transactions on Evolutionary Computation* **10**(6), 646–657.

Castro, L. N. D.: 2002, *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer-Verlag, London.

Chakraborti, N., Misra, K., Bhatt, P., Barman, N. and Prasad, R.: 2001, Tight-binding calculations of si-h clusters using genetic algorithms and related techniques: Studies using differential evolution, *Journal of Phase Equilibria and Diffusion* **22**, 525–530.

DaCosta, L., Fialho, A., Schoenauer, M. and Sebag, M.: 2008, Adaptive operator selection with dynamic multi-armed bandits, *Proceedings of the 10th annual conference on Genetic and evolutionary computation (GECCO 2008)*, ACM, pp. 913–920.

Darwin, C.: 1859, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*.

Das, S. and Konar, A.: 2005, An improved differential evolution scheme for noisy optimization problems, *Proceedings of the First international conference on Pattern Recognition and Machine Intelligence*, PReMI'05, Springer-Verlag, Berlin, Heidelberg, pp. 417–421.

Davis, L.: 1989, Adapting operator probabilities in genetic algorithms, *Proceedings of the third international conference on Genetic algorithms*, Morgan Kaufmann Publishers Inc., pp. 61–69.

Eiben, G. and Schut, M. C.: 2008, New ways to calibrate evolutionary algorithms, *Advances in Metaheuristics for Hard Optimization*, pp. 153–177.

Epitropakis, M., Plagianakos, V. and Vrahatis, M.: 2009, Evolutionary adaptation of the differential evolution control parameters, *IEEE Congress on Evolutionary Computation, 2009. CEC 2009*, Trondheim, Norway, pp. 1359–1366.

Fialho, A., Costa, L., Schoenauer, M. and Sebag, M.: 2008, Extreme value based adaptive operator selection, *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, Springer-Verlag, Berlin, Heidelberg, pp. 175–184.

Fialho, A., Costa, L., Schoenauer, M. and Sebag, M.: 2009, Learning and intelligent optimization, Springer-Verlag, chapter Dynamic Multi-Armed Bandits and Extreme Value-Based Rewards for Adaptive Operator Selection in Evolutionary Algorithms, pp. 176–190.

Fialho, A., Ros, R., Schoenauer, M. and Sebag, M.: 2010, Comparison-based adaptive strategy selection with bandits in differential evolution, *Proceedings of the 11th international conference on Parallel problem solving from nature: Part I*, PPSN'10, Springer-Verlag, Berlin, Heidelberg, pp. 194–203.

Fialho, A., Schoenauer, M. and Sebag, M.: 2009, Analysis of adaptive operator selection techniques on the royal road and long k-path problems, *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, ACM, New York, NY, USA, pp. 779–786.

Finck, S., Hanseny, N., Raymond, R. and Augerx, A.: 2010, Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions, *Working Paper 2009/20, compiled March 24, 2012* pp. 1–102.

Freitas, A. R. R., Pedrosa Silva, R. C. and Guimarães, F. G.: 2012, Differential evolution and perceptron decision trees for fault detection in power transformers, *Soft Computing Models in Industrial and Environmental Applications*, Vol. 188 of *Advances in Intelligent Systems and Computing*, Springer Berlin Heidelberg, pp. 143–152.

Gämperle, R., Müller, S. D. and Koumoutsakos, P.: 2002, A parameter study for differential evolution, *WSEAS International Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Press, pp. 293–298.

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1 edn, Addison-Wesley Professional.

Gong, W., Fialho, A. and Cai, Z.: 2010, Adaptive strategy selection in differential evolution, *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, ACM, New York, NY, USA, pp. 409–416.

Guimarães, F. G.: 2009, *Evolução Diferencial*, Manual de Computação Evolutiva e Metaheurística, Imprensa da Universidade de Coimbra/Editora da Universidade Federal de Minas Gerais, pp. 141–162.

Guo, M., Liu, Y. and Malec, J.: 2004, A new q-learning algorithm based on the metropolis criterion, *IEEE Transactions on Systems, Man, and Cybernetics, Part B* pp. 2140–2143.

Julstrom, B. A.: 1995, What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm, *Proceedings of the 6th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., pp. 81–87.

Kaelbling, L. P., Littman, M. L. and Moore, A. W.: 1996, Reinforcement learning: a survey, *Journal of Artificial Intelligence Research* **4**, 237–285.

Karaboga, N.: 2005, Digital IIR filter design using differential evolution algorithm, *EURASIP J. Appl. Signal Process.* **2005**, 1269–1276.

Kennedy, J. and Eberhart, R. C.: 1995, Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.: 1983, Optimization by simulated annealing, *Science, Number 4598, 13 May 1983* **220, 4598**, 671–680.

Lampinen, J. and Zelinka, I.: 2000, On stagnation of the differential evolution algorithm, *Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing*, pp. 76–83.

Li, K., Fialho, A. and Kwong, S.: 2011, Multi-objective differential evolution with adaptive control of parameters and operators, *Proceedings of the 5th international conference on Learning and Intelligent Optimization*, LION'05, Springer-Verlag, Berlin, Heidelberg, pp. 473–487.

Liu, J. and Lampinen, J.: 2002, On setting the control parameter of the differential evolution method, *8th International Conference on Soft Computing (MENDEL2002)*, pp. 11–18.

Liu, J. and Lampinen, J.: 2005, A fuzzy adaptive differential evolution algorithm, *Soft Computing* **9**, 448–462.

Liu, X., Shi, L. and Chen, R.: 2011, Jdel: Differential evolution with local search mechanism for high-dimensional optimization problems, *in* L. Qi (ed.), *Information and Automation*, Vol. 86 of *Communications in Computer and Information Science*, Springer Berlin Heidelberg, pp. 347–352.

Lobo, F. G., Lima, C. F. and Michalewicz, Z.: 2007, *Parameter Setting in Evolutionary Algorithms*, 1st edn, Springer Publishing Company, Incorporated.

Lopes, R., Freitas, A., Pedosa Silva, R. and Guimarães, F.: 2012, Differential evolution and perceptron decision trees for classification tasks, *in* H. Yin, J. A. Costa and G. Barreto (eds), *Intelligent Data Engineering and Automated Learning - IDEAL 2012*, Vol. 7435 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 550–557.

Maturana, J., Fialho, Á., Saubion, F., Schoenauer, M. and Sebag, M.: 2009, Extreme Compass and Dynamic Multi-Armed Bandits for Adaptive Operator Selection,

IEEE Congress on Evolutionary Computation (CEC), IEEE, Trondheim, Norvège, pp. 365–372.

Maturana, J. and Saubion, F.: 2008, A compass to guide genetic algorithms, *Proceedings of the 10th international conference on Parallel Problem Solving from Nature: PPSN X*, Springer-Verlag, pp. 256–265.

Mezura-Montes, E., Velázquez-Reyes, J. and Coello Coello, C. A.: 2006, A comparative study of differential evolution variants for global optimization, *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ACM, New York, NY, USA, pp. 485–492.

Montgomery, D.: 2008, *Design and Analysis of Experiments*, John Wiley & Sons.

Onwubolu, G. C. and Davendra, D.: 2006, Scheduling flow shops using differential evolution algorithm, *European Journal of Operational Research* **171**(2), 674–692.

Pant, M., Ali, M. and Abraham, A.: 2009, Mixed mutation strategy embedded differential evolution, *Proceedings of the Eleventh conference on Congress on Evolutionary Computation - CEC'09*, IEEE Press, Piscataway, NJ, USA, pp. 1240–1246.

Pedrosa Silva, R. C., Lopes, R. A. and Guimarães, F. G.: 2011, Self-adaptive mutation in the differential evolution, *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11*, ACM, New York, NY, USA, pp. 1939–1946.

Prado, R. S., Pedrosa Silva, R. C., Guimarães, F. G. and Neto, O. M.: 2010, A new differential evolution based metaheuristic for discrete optimization., *Intenational Journal of Natural Computing Research (IJNCR)* **1**(2), 15–32.

Price, K. V., Storn, R. M. and Lampinen, J. A.: 2005, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer-Verlag.

Qin, A. K., Huang, V. L. and Suganthan, P. N.: 2009, Differential evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation* **13**(2), 398–417.

Qin, A. K. and Suganthan, P. N.: 2005, Self-adaptive differential evolution algorithm for numerical optimization, *Congress on Evolutionary Computation'05*, pp. 1785–1791.

Reddy, M. J. and Kumar, D. N.: 2007, Multiobjective differential evolution with application to reservoir system optimization, *Journal of Computing in Civil Engineering* **21(2)**, 136–146.

Sai, H. A., Vinaya, B. A., Govardhan, A. and Satapathy, S. C.: 2010, Data clustering using almost parameter free differential evolution technique, *International Journal of Computer Applications* **8**(13), 1–7.

Salman, A., Engelbrecht, A. P. and Omran, M. G.: 2006, Empirical analysis of self-adaptive differential evolution, *European Journal of Operational Research* **In Press, Corrected Proof**.

Schwefel, H.: 1981, *Numerical optimization of computer models*, Wiley, Chichester, WS, UK.

Storn, R. and Price, K.: 1995, Differential evolution- a simple and efficient adaptive scheme for global optimization over continuous spaces, *Technical report*.

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

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

Thierens, D.: 2005, An adaptive pursuit strategy for allocating operator probabilities, *Proceedings of the 2005 conference on Genetic and evolutionary computation - GECCO '05* p. 1539.

Wang, X.-z., Li, P. and Yu, G.-y.: 2010, Multi-objective hybrid differential evolution algorithm based on pareto optimal solution migration, *Proceedings of the 2010 International Conference on Computational Aspects of Social Networks*, IEEE Computer Society, Washington, DC, USA, pp. 547–551.

Wang, Y., Cai, Z. and Zhang, Q.: 2011, Differential evolution with composite trial vector generation strategies and control parameters, *IEEE Trans. Evolutionary Computation* pp. 55–66.

Watkins, C. J. C. H.: 1989, *Learning from Delayed Rewards*, PhD thesis, King's College.

Watkins, C. J. C. H. and Dayan, P.: 1992, Technical note: Q-learning, *Machine Learning* **8**(3), 279–292.

Whitacre, J. M., Pham, T. Q. and Sarker, R. A.: 2006, Use of statistical outlier detection method in adaptive evolutionary algorithms, *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO '06, ACM, New York, NY, USA, pp. 1345–1352.

Wineberg, M. and Oppacher, F.: 2003, The underlying similarity of diversity measures used in evolutionary computation, *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII*, GECCO'03, Springer-Verlag, Berlin, Heidelberg, pp. 1493–1504.

Wunder, M., Littman, M. L. and Babes, M.: 2010, Classes of multiagent q-learning dynamics with epsilon-greedy exploration, *ICML*, pp. 1167–1174.

Yuan, B. and Gallagher, M.: 2004, Statistical racing techniques for improved empirical evaluation of evolutionary algorithms, *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, Springer, pp. 172–181.

Zaharie, D.: 2002, Critical values for the control parameters of differential evolution algorithm, *in* R. Matouek and P. Omera (eds), *Proceedings of MENDEL 2002, 8th International Mendel Conference on Soft Computing, June 5.7.2002, Brno, Czech Republic*.

Zaharie, D.: 2009, Influence of crossover on the behavior of differential evolution algorithms., *Applied Soft Computing* **9**(3), 1126–1138.

Zamuda, A., Brest, J., Boskovic, B. and Zumer, V.: 2009, Differential Evolution with Self-Adaptation and Local Search for Constrained Multiobjective Optimization, *2009 IEEE Congress on Evolutionary Computation (CEC'2009)*, IEEE Service Center, Trondheim, Norway, pp. 195–202.

Zhang, J. and Sanderson, A. C.: 2007, Jade: Self-adaptive differential evolution with fast and reliable convergence performance., *IEEE Congress on Evolutionary Computation*, IEEE, pp. 2251–2258.

Zhang, J. and Sanderson, A. C.: 2009, Jade: adaptive differential evolution with optional external archive, *Transactions on Evolutionary Computation* **13**(5), 945–958.

Zhang, W. J. and Xie, X. F.: 2003, DEPSO: hybrid particle swarm with differential evolution operator, *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, Vol. 4, pp. 3816–3821.