Métodos Computacionais Inspirados na Natureza - Relatório da Fase 1

Lucas de Carvalho Gomes (DRE: 120175208)
Universidade Federal do Rio de Janeiro - COPPE - Rio de Janeiro, Brazil
lucas.gomes@coppe.ufrj.br

***Resumo*—Com base nos algoritmos apresentados nas aulas da disciplina, foram efetuados ajustes de modo a aprimorar seus desempenhos. As versões ajustadas dos algoritmos foram testadas nas cinco primeiras funções usadas como *benchmarks* para a edição de 2005 do IEEE CEC (*Congress on Evolutionary Computation*). Os resultados indicam diferenças importantes no comportamento de cada algoritmo, em termos de velocidade de convergência, capacidade de escapar de mínimos locais, resistência a ruídos, robustez quanto aos parâmetros utilizados e facilidade de utilização.**

I. Introdução

Durante as aulas da disciplina de Métodos Computacionais Inspirados na Natureza, foram apresentados sete algoritmos: Algoritmos Genéticos (ou GA) [1]–[3], Evolução Diferencial (ou DE – *Differential Evolution*) [4], Otimização de Enxame de Partículas (ou *Particle Swarm Optimization* – PSO) [5]–[7], Algoritmos de Aranhas Sociais, dos quais o autor escolheu, conforme acordado em sala de aula, o *Social Spider Optimization* (ou SSO) [8], Colônia Artificial de Abelhas (ou *Artificial Bee Colony* – ABC) [9] e a extensão do algoritmo de Otimização de Colônias de Formigas (ou *Ant Colony Optimization* – ACO) [10] para otimização numérica.

Este relatório tem como objetivo apresentar o desempenho destes algoritmos sobre cinco das funções de teste do IEEE CEC 2005 (*Congress on Evolutionary Computation*) [11]. O objetivo, para todas as funções, é a minimização dos seus valores. Para isto, os parâmetros dos algoritmos foram ajustados para encontrar valores que se ajustem melhor às funções de *benchmark*. Por fim, também foi testada uma versão adaptativa do Algoritmo Genético, com uma alteração na mutação.

Todos os algoritmos foram implementados pelo autor em Python. A implementação, bem como os arquivos de resultados, estão localizados no repositório de código aberto [12].

O restante do relatório está organizado da seguinte forma. A Seção II discute as configurações dos algoritmos utilizados, bem como fornece uma breve descrição das variantes extras testadas. A Seção III descreve a configuração dos experimentos realizados para verificar a eficácia nas funções de teste do CEC, enquanto a Seção IV mostra os resultados destes experimentos. Por fim, a Seção V conclui o trabalho com uma discussão sobre os resultados encontrados.

II. Configurações dos Algoritmos Utilizados

A seguir, as especificações são descritas algoritmo a algoritmo.

A. Algoritmo Genético

O GA foi utilizado na sua implementação padrão, conforme descrito em Man *et al.* [1], e Eiben e Smith [2]. Cada um dos seus componentes é descrito a seguir:

- População: tamanho 100; inicializada aleatoriamente com distribuição uniforme;
- Seleção de pais: torneio. A quantidade de competidores por rodada é 2. A quantidade de pais selecionados ao fim do torneio é 100;
- Recombinação: BLX-α, tal como proposto por [3]. O valor adotado para a constante α é de $0,5$ e a probabilidade de *crossover* usada foi 1. Cada operação de *crossover* gera 1 filho.
- Mutação: foi escolhida a mutação uniforme. Dada a sua força, por gerar um gene novo totalmente aleatório, foi escolhido um valor baixo de probabilidade de mutação: 0,05;
- Seleção de sobreviventes: foi usado o esquema referido por Eiben e Smith [2] como GENITOR. A população da geração atual (100 indivíduos) e os filhos gerados nesta geração (outros 100 indivíduos) são juntados e ordenados em ordem decrescente de aptidão. Por fim, os 100 melhores indivíduos são mantidos. Este mecanismo foi utilizado para aumentar a pressão de seleção. A quantidade de pais e filhos é responsável por aumentar a diversidade na população.
- Tratamento de indivíduos que violam os intervalos: reexecução da operação em questão (mutação ou recombinação).

B. Algoritmo Genético Adaptativo

Uma modificação foi realizada no GA tradicional, seguindo o exposto por Eiben e Smith [2]. O arranjo descrito nesta referência altera a operação de mutação, para que ela se adapte ao desempenho atual do algoritmo e ao comportamento da função. Nesta modificação, a mutação utilizada é a *creep*, que gera perturbações normais. A média de tais perturbações foi mantida como 0.

A modificação proposta altera os desvios-padrão em cada dimensão a cada rodada. Para isto, estes valores são adicionados aos genótipos dos indivíduos e sofrem alterações pelo *crossover*. Antes de cada fase de mutação, eles são alterados, seguindo as seguintes equações:

$$\sigma_i' = \sigma_i \times e^{\tau' N(0,1) + \tau N_i(0,1)}, \quad (1)$$

sendo σ_i o desvio-padrão da dimensão i, $sigma'_i$ seu próximo valor, τ' um parâmetro que ajusta as perturbações de todas as dimensões, τ é um parâmetro que ajusta as perturbações individuais em cada dimensão, N é uma perturbação normal global e N_i é uma perturbação normal em cada dimensão. Com os valores calculados de desvio-padrão, as dimensões do genótipo de um novo indivíduo são calculadas da seguinte forma:

$$x'_i = x_i + \sigma'_i \times N_i(0,1) \quad (2)$$

Os valores adotados para τ' e τ foram, respectivamente, $1/\sqrt{(2n)}$ e $1/\sqrt{(2\sqrt{(n)})}$. Para evitar que os desvios-padrão se aproximem demais de 0, o limite inferior $\epsilon_0 = 0,1$ foi utilizado. Caso algum valor σ_i calculado seja menor que ϵ_0, o valor de ϵ_0 é atribuído a σ_i. O valor de ϵ_0 foi obtido através de execuções sucessivas do algoritmo e comparações preliminares.

Todas as outras configurações deste algoritmo genético são iguais à sua implementação tradicional, discutida na última subseção.

C. Evolução Diferencial

A implementação proposta por Storn e Price [4] foi implementada sem modificações. Foi escolhida a sua variante padrão, o DE $rand/1/bin$, usando na mutação 1 vetor de diferença, o parâmetro F (força de mutação) igual a 0,5 e a constante de crossover (CR) também igual a 0,5. A quantidade de indivíduos da população foi 100. Além da implementação padrão, duas decisões de implementação foram tomadas: inicializar a população com valores aleatórios vindos de uma distribuição uniforme; e, assim como no GA, ao se encontrar indivíduos que violam os intervalos, tais indivíduos são gerados novamente pela mesma operação, até que um indivíduo válido seja gerado.

D. Otimização de Enxame de Partículas

A implementação padrão, descrita por Kennedy e Eberhart [5], [6], foi implementada. No entanto, alteração foi efetuada no PSO tradicional, seguindo o exposto por Shi e Eberhart [7]. Os parâmetros do algoritmo são apresentados a seguir:

- População: 80 indivíduos; início aleatório por distribuição uniforme;
- Pesos das componentes local e global: dadas as conclusões de Kennedy e Eberhart [5], [6] de que um desempenho melhor é obtido ao se usar valores idênticos para as influências local e global, os valores de c_1 e c_2, correspondentes a estas influências, foram configurados para o valor $2,05$;
- Decaimento de inércia: foi utilizada a modificação proposta por Shi e Eberhart [7]: uma variável é adicionada para multiplicar a velocidade anterior. Este fator de inércia foi configurado para variar entre 0,9 e 0,4, diminuindo ao longo da execução do algoritmo;
- Velocidade máxima: configurada como 10 (número de dimensões das funções consideradas);
- Tratamento de indivíduos que ultrapassam os limites dos intervalos: gerar uma variável aleatória por uma distribuição uniforme, com os extremos sendo os limites do intervalo, na posição da violação;

E. Aranhas Sociais

O algoritmo de Aranhas Sociais escolhido foi o *Social Spider Optimization*, proposto por Cuevas *et al.* [8]. No entanto, algumas modificações foram necessárias para que ele convergisse. Ao tentar reproduzir os resultados dos autores para as mesmas funções, verificou-se que implementar o algoritmo somente com as equações apresentadas não era o suficiente para alcançar a convergência. Para melhorar o desempenho e possibilitar a reprodução dos resultados dos autores, duas mudanças foram efetuadas no algoritmo:

- As distâncias entre as aranhas foram suavizadas por um procedimento de normalização. Desta forma, evitou-se que as funções exponenciais que usam os negativos de seus quadrados como expoentes convergissem rapidamente para 0, o que ocorreu para todos os casos com os intervalos [-100, +100] considerados nas funções tratadas neste trabalho. Este fator foi o que impediu a convergência. A normalização consiste na seguinte equação:

  $$d'_{ab} = d_{ab}/D, \quad (3)$$

  onde d'_{ab} é a distância normalizada entre duas aranhas a e b, d_{ab} é o valor real desta distância e D é o diâmetro do intervalo de busca. O diâmetro é calculado da seguinte forma:

  $$D = ||\mathbf{x}_{max} - \mathbf{x}_{min}||_2, \quad (4)$$

  onde $\mathbf{x}_{max}$ representa uma solução com todos os valores das variáveis de decisão x_i situados nos extremos superiores de seus intervalos, e $\mathbf{x}_{min}$, de maneira análoga, é uma solução com todos os seus valores de x_i situados nos extremos inferiores.
- Foi adicionado um critério guloso que acelerou a convergência do algoritmo: a implementação padrão faz todas as aranhas se moverem, independentemente da qualidade de suas soluções. A modificação é fazer com que a aranha que possui a melhor solução em uma geração não se mova naquela geração. Deste modo, soluções de boa qualidade são preservadas e, se esta solução se manter a melhor, outras aranhas podem se mover em direção a ela e realizar buscas locais aleatórias, realizando explotações. Os parâmetros escolhidos foram os sugeridos pelos criadores do algoritmo: a população foi constituída de 30 aranhas, e a probabilidade de aranhas fêmeas realizarem movimentos de atração (valor definido como PF) foi estabelecida como $0,7$. As aranhas foram inicializadas, como em todos os outros algoritmos, por uma distribuição uniforme. O tratamento de indivíduos que violam o domínio das variáveis usado aqui é diferente do usado nos algoritmos anteriores: as variáveis cujos valores violam um dos dois extremos têm seus valores ajustados para o

valor do extremo mais próximo, efetivamente truncando a variável de decisão.

F. Colônia Artificial de Abelhas

O algoritmo foi implementado conforme especificado por [9], sem modificações. Ou seja, há tantas abelhas operárias quanto observadoras e apenas 1 abelha batedora – que é responsável pela busca aleatória – é gerada a cada iteração. A população foi configurada para ter um total de 50 indivíduos, sendo 25 operárias e 25 observadoras. O valor do limite para a melhoria de uma solução é de 150 iterações. Como no algoritmo SSO, indivíduos que violam os intervalos do domínio são truncados.

G. Otimização de Colônias de Formigas

A extensão do algoritmo ACO para problemas de otimização numética foi implementada tal como descrito pelo trabalho de Socha e Dorigo [10]. Os parâmetros utilizados foram os sugeridos pelos autores, sem modificações. Ou seja: foram utilizadas 2 formigas, um arquivo de 50 soluções, uma constante de velocidade de convergência (ξ) de $0,85$ e uma constante de localidade de busca (q) de 10^{-4}. Da mesma forma que nos dois trabalhos anteriores, indivíduos que violam os intervalos são truncados.

III. Experimentos

Conforme mencionado na Seção I, as funções que serão testadas neste trabalho são as cinco primeiras funções de teste unimodais do CEC 2005, descritas em um documento de Suganthan *et al.* [11]. Para facilitar a implementação destas funções, foi instalada a biblioteca `optproblems` [13], que é programada em Python e é de código aberto, contendo as funções do CEC 2005 implementadas e seus parâmetros de desvio (shift). Foram utilizadas as versões com 10 dimensões das funções. Todas as funções têm o espaço de busca limitado ao intervalo $[-100, +100]$ para todas as variáveis.

Como requisitado na proposta do trabalho, há dois critérios de parada para todos os algoritmos: a execução de $10000 \times D$ avaliações da função objetivo (FES – *Fitness Evaluations*), onde D é o número de dimensões, e a obtenção de um erro de 10^-8 ou menor, em relação ao valor ótimo conhecido das funções.

Todos os algoritmos foram executados 25 vezes para cada função.

IV. Resultados

As Tabelas I a VII apresentam os valores de erro para cada algoritmo. As colunas apresentam, em ordem, a função avaliada, o melhor valor do erro, o pior valor de erro, a mediana dos erros, a média dos erros, o desvio-padrão deles e a taxa de sucesso do algoritmo (fração de vezes em que o algoritmo atingiu a precisão requerida). Para facilitar a visualização, todos os valores exibidos são arrendodados para três casas decimais. Caso o leitor considere necessário, os arquivos originais que geraram estas tabelas estão disponíveis no repositório público do GitHub com as implementações do autor [12], no diretório denominado *Results*.

As Figuras 1 a 15 apresentam os gráficos de convergência para todas as funções. Cada gráfico apresenta, sobrepostos, os resultados de cada algoritmo. O eixo horizontal apresenta a quantidade de avaliações da função objetivo; os números apresentados nos eixos são multiplicadores do limite de avaliações da função objetivo (MaxFES). Os pontos gerados correspondem aos valores de FES critérios definidos para o CEC 2005, ou seja, o conjunto $\{0,0; 0,001; 0,01; 0,1; 0,2; ...; 0,9; 1,0\}$. O eixo vertical apresenta o erro da melhor solução presente na geração atual. Esta medida de erro foi escolhida no lugar do erro da melhor solução global encontrada porque é possível observar se algum algoritmo descarta boas soluções durante a sua execução. Dado que a escala do erro pode ser muito alta, alcançando valores superiores a 10^6 para algumas funções, foi usada uma escala logarítmica: calculou-se o logaritmo natural na base 10 de todos os valores de erro, acrescidos de 1, dado que os valores de erro podem alcançar 0, o que inviabilizaria o cálculo do logaritmo. Ou seja, o erro representado no gráfico é:

$$\epsilon_{graf} = ln(1 + \epsilon_{real}), \tag{5}$$

sendo ϵ_{graf} o erro representado no gráfico e ϵ_{real} o erro real.

A seguir, serão comentados os resultados por função:

- F1: a esfera, por apresentar derivadas parciais lineares em relação a cada uma de suas variáveis, não apresentar descontinuidades e ruído, é a função mais simples do conjunto, de modo que todos os algoritmos, exceto o SSO, apresentam uma taxa de sucesso de 100%. Os algoritmos testados, incluindo o SSO, convergem rapidamente para perto do ótimo local, requerendo, no máximo, 30% das FES para alcançar a convergência.
- F2: nesta, apenas o GA adaptativo, o DE, o PSO e o ACO obtêm sucesso, sendo que o GA adaptativo obtém uma taxa de sucesso de 76% enquanto os restantes obtém 100% de sucesso. Esta função, como indicam os resultados, é mais desafiadora, por conta da presença de mínimos locais em sua superfície e uma taxa de decréscimo menor. Isso diminuiu a velocidade de convergência dos algoritmos e inviabilizou a convergência do GA convencional e do ABC.
- F3: esta função se mostrou ser a mais difícil de otimizar. Apenas o DE conseguiu obter sucesso, tendo uma taxa de 100%. Além disto, os valores de erro obtidos pelos outros algoritmos possuem escalas altas, chegando a, no máximo, 7×10^6 – pior valor de erro para o SSO. Estes resultados estão relacionados com a rotação, a alta escala dos valores assumidos pela função e pela presença de mínimos locais e regiões de baixa taxa de decréscimo;
- F4: esta função é a F2 adicionada de ruído, sendo a única função que possui ruído dentre as analisadas. A adição de ruído prejudicou o desempenho dos algoritmos, pois afetou a taxa de variação observada com o movimento das soluções que formam a população. Os algoritmos que conseguiram encontrar o ótimo global foram o GA adaptativo, o DE, o PSO e o ACO. No entanto, o GA adaptativo e o PSO apresentam taxas muito baixas de

sucesso – respectivamente, 12% e 8% –, um indicativo da sensibilidade desses dois algoritmos ao ruído.

- F5: esta função possui mínimos locais e o ótimo global nas bordas do espaço de busca. Portanto, algoritmos que truncam um indivíduo que viola as bordas ou refletem o passo que violou as bordas, no lugar de gerar novos valores aleatórios, podem ter uma vantagem na otimização da F5. Os algoritmos que usaram o truncamento de indivíduos foram o SSO, o ABC e o ACO. Esta função mostrou ter uma dificuldade similar à F3, dado que, da mesma forma que com ela, só um algoritmo obteve sucesso na F5: o ACO. Dos algoritmos que não obtiveram sucesso, o que conseguiu se aproximar mais foi o GA adaptativo, com o melhor erro sendo de 10^{-4}.

Agora, cada algoritmo será comentado individualmente:

- GA: o Algoritmo Genético é um algoritmo que tem um grande potencial de personalização, dada a grande quantidade de representações de indivíduos e técnicas de crossover, mutação e seleção de sobreviventes. No entanto, ele é bem sensível a estas técnicas e aos seus parâmetros. Uma implementação convencional do GA para valores reais obtém sucesso limitado na otimização de funções para algumas implementações, requerendo experiência e técnicas adicionais para obter resultados competitivos.
- GA adaptativo: o acréscimo de um parâmetro de adaptação na mutação conseguiu aprimorar o desempenho do GA convencional, permitindo que ele encontrasse o ótimo em mais duas funções, F2 e F4, além da F1 que o GA convencional já conseguia otimizar. Além disso, ela se aproxima do valor ótimo da F5, obtendo uma precisão de 10^{-4}. No entanto, ele apresenta uma grande sensibilidade a ruído, como é possível perceber ao se comparar os resultados da F2 e da F4. Portanto, para melhorar o seu resultado neste *benchmark*, modificações adicionais precisam ser adotadas.
- DE: a Evolução Diferencial é um algoritmo de baixa complexidade, e se mostrou muito robusto para resolver a maioria das funções do *benchmark*, obtendo 100% de sucesso para F1, F2, F3 e F4, usando a sua variante mais simples, $rand/1/bin$. Assim, ele é um algoritmo de aplicação mais fácil. No entanto, em experimentos preliminares, não registrados nas tabelas e nos gráficos, a DE apresenta uma sensibilidade quanto as parâmetros utilizados, sobretudo a constante de *crossover* CR. Para aprimorar ainda mais o seu desempenho, pode ser interessante investigar técnicas adaptativas.
- PSO: o PSO apresentou desempenho similar ao do GA adaptativo, conseguindo encontrar o ótimo nas funções 1, 2 e 4. No entanto, a baixa taxa de sucesso na F4 demonstra uma grande sensibilidade ao ruído. Em experimentos preliminares, não registrados nos resultados exibidos, o uso do fator de decaimento de inércia se mostrou uma modificação interessante, melhorando os resultados. O algoritmo também se mostrou sensível a modificações na velocidade máxima e aos pesos dos componentes pessoal e global (*gbest* e *pbest*). Como indicado por alguns trabalhos, o ajuste desses parâmetros depende da função que visa-se otimizar, podendo ser interessante desequilibrar os pesos de *gbest* e *pbest* [6], [14]. Investigações com outros valores para os parâmetros e técnicas para aprimorar a busca podem ser úteis para efetuar melhorias.
- SSO: O SSO foi o único algoritmo que não conseguiu atingir a precisão desejada para nenhuma função, mesmo com as modificações realizadas e mesmo conseguindo-se reproduzir os resultados dos autores do algoritmo ao se utilizar os parâmetros usados pelos autores [8]. No entanto, ele converge para valores próximos do ótimo para as funções 1, 2 e 4, embora na F4 haja um valor alto de desvio padrão $(270, 304)$. Apesar deste desempenho inferior ao de outros algoritmos nesta rodada de testes, vale observar que, como mostrado no artigo que originou o algoritmo [8], ele apresenta um desempenho mais resistente à dimensionalidade do problema. Para a próxima fase do trabalho, visto que as funções analisadas terão trinta dimensões – a quantidade usada nos testes dos autores –, este algoritmo pode se mostrar uma alternativa interessante, sobretudo se for encontrado um bom conjunto de parâmetros e modificações adicionais.
- ABC: o ABC apresentou uma taxa de convergência rápida para a F1. Como é possível ver pela Figura 2, no melhor caso, ele consegue convergir mais rápido para esta função do que outros algoritmos. Isto indica que ele possui um bom potencial para explotação, algo que é muito reforçado pelo algoritmo pelo uso das abelhas observadoras, que escolhem com maior prioridade soluções com melhor aptidão para efetuar melhorias. No entanto, a busca realizada pelo ABC, por usar um termo de limite de melhorias efetuadas, que, ao violado, causa a troca de uma solução por uma nova solução aleatória, pode fazer o algoritmo ter dificuldades de convergência, como também enfatizado por Cuevas *et al.* [8]. Isto pode ser observado nas Figuras 4, 5, 6, 7, 8, 9, 11 e 12. Nestas Figuras, há uma piora da melhor solução da população. Por conta disto, para usar o ABC, deve-se ponderar com cautela o uso do termo de limite e a quantidade de abelhas batedoras, que efetuam buscas aleatórias após esse limite ser ultrapassado.
- ACO: esse algoritmo é uma adaptação de um algoritmo para otimização combinatória, o que normalmente gera algoritmos de precisão menor que algoritmos voltados para otimização numérica, como sinalizado pelos próprios criadores do algoritmo, [10]. No entanto, ele apresentou um desempenho robusto para a otimização de quase todas as funções, obtendo uma taxa de sucesso de 100% em todas as funções, com exceção da F3, além de uma velocidade de convergência alta e competitiva. Em termos da quantidade de funções otimizadas e da taxa de sucesso total, ele está empatado com o DE. Ele também se mostrou um algoritmo de fácil utilização: os

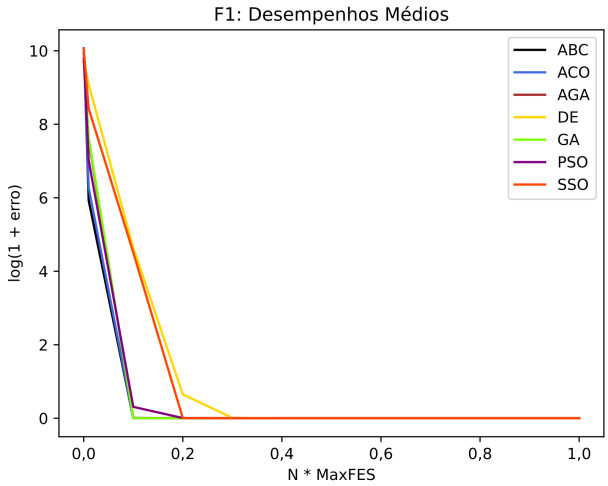

Figura 1: Gráficos de convergência para a F1 - média das soluções.

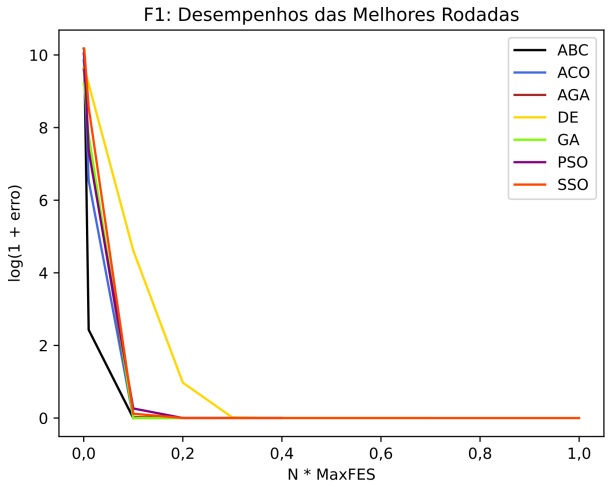

Figura 2: Gráficos de convergência para a F1 - melhor rodada.

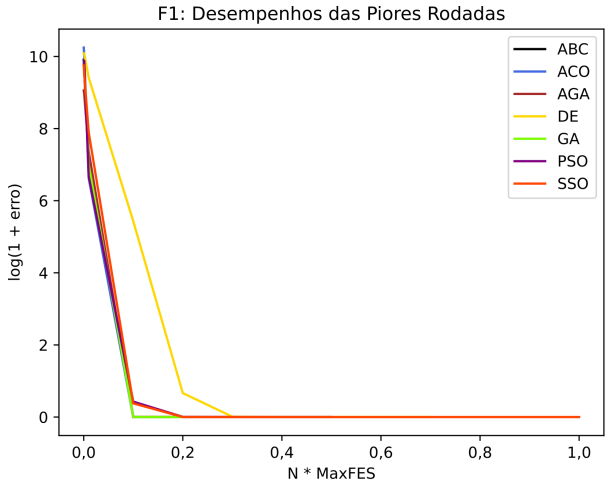

Figura 3: Gráficos de convergência para a F1 - pior rodada.

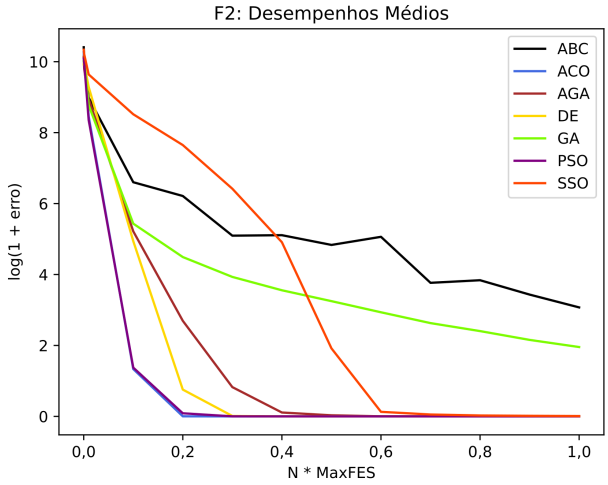

Figura 4: Gráficos de convergência para a F2 - média das soluções.

parâmetros sugeridos pelos autores, sem modificações, resultaram nos resultados robustos mostrados nos gráficos de convergência e na Tabela VII. Com este desempenho, este algoritmo pode se mostrar interessante para o uso em funções mais desafiadoras, como as da Fase 2 do trabalho. No entanto, deve-se verificar os fatores que fizeram este algoritmo não executar tão bem na F3. Possivelmente, hibridizar este algoritmo com o DE poderá trazer bons resultados.

V. Conclusão

Este relatório realizou testes com 6 algoritmos, além da adaptação de 1 deles – o GA adaptativo –, para a otimização de funções de minimização de teste do CEC 2005. Por meio da execução de 25 rodadas e da análise das tabelas de resultados e dos gráficos de convergência, pôde-se extrair diferentes características dos algoritmos estudados em sala de aula. Alguns algoritmos, para serem utilizados, requerem maior cuidado com os parâmetros e componentes usados – Algoritmo Genético, Otimização de Enxame de Partículas e Colônia de Abelhas Artificial são exemplos. Outros apresentam uma facilidade maior de se empregar, apresentando eficiência e robustez nas variantes padrão e com parâmetros padrão, sem técnicas de otimização ou adaptação – DE e ACO. O uso de técnicas adaptativas apresenta um potencial importante que pode ser aplicado a qualquer algoritmo. Como os resultados indicaram, o GA obteve um grande benefício ao usar a adaptação na mutação *creep*. Por outro lado, um algoritmo que apresenta resultados robustos para problemas de alta dimensionalidade, o SSO, não apresenta resultados tão precisos em problemas de baixa dimensionalidade. Vale observar, na fase 2, a sensibilidade dos algoritmos restantes à dimensionalidade dos problemas. O conhecimento destas características de cada algoritmo e de técnicas eficientes de melhorias pode tornar possível a criação de algoritmos híbridos, por meta-otimização

Tabela I: Resultados para o GA convencional.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$3{,}025\times10^{-9}$	$9{,}9635\times10^{-9}$	$8{,}7\times10^{-9}$	$8{,}025\times10^{-9}$	$1{,}923\times10^{-9}$	100%
2	0,527	21,156	4,025	6,063	5,4	0%
3	50550,729	1269388,363	403487,446	435369,54	327863,282	0%
4	2,179	208,639	20,006	40,339	46,069	0%
5	53,504	1509,116	354,012	409,919	347,393	0%

Tabela II: Resultados para o GA adaptativo.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$4{,}569\times10^{-9}$	$9{,}998\times10^{-9}$	$8{,}145\times10^{-9}$	$7{,}936\times10^{-9}$	$1{,}582\times10^{-9}$	100%
2	$5{,}203\times10^{-9}$	$1{,}152\times10^{-7}$	$9{,}519\times10^{-9}$	$1{,}572\times10^{-8}$	$2{,}183\times10^{-8}$	76%
3	15155,861	391046,694	101802,904	134683,0302	98302,151	0%
4	$5{,}834\times10^{-9}$	110,137	$3{,}675\times10^{-5}$	5,393	22,201	12%
5	0,0001	544,421	0,0004	26,460	108,561	0%

Tabela III: Resultados para o DE.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$4{,}775\times10^{-9}$	$9{,}9195\times10^{-9}$	$8{,}109\times10^{-9}$	$8{,}021\times10^{-9}$	$1{,}275\times10^{-9}$	100%
2	$4{,}187\times10^{-9}$	$9{,}979\times10^{-9}$	$8{,}13\times10^{-9}$	$7{,}778\times10^{-9}$	$1{,}808\times10^{-9}$	100%
3	$3{,}581\times10^{-9}$	$9{,}993\times10^{-9}$	$8{,}621\times10^{-9}$	$8{,}106\times10^{-9}$	$1{,}835\times10^{-9}$	100%
4	$3{,}845\times10^{-9}$	$9{,}824\times10^{-9}$	$8{,}023\times10^{-9}$	$7{,}801\times10^{-9}$	$1{,}538\times10^{-9}$	100%
5	71,809	3125,9	826,347	1293,354	1010,892	0%

Tabela IV: Resultados para o PSO.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$7{,}328\times10^{-9}$	$9{,}943\times10^{-9}$	$9{,}081\times10^{-9}$	$8{,}97\times10^{-9}$	$6{,}277\times10^{-10}$	100%
2	$7{,}618\times10^{-9}$	$9{,}842\times10^{-9}$	$9{,}277\times10^{-9}$	$9{,}057\times10^{-9}$	$7{,}008\times10^{-10}$	100%
3	44,551	1032752,809	3354,755	132004,333	285816,417	0%
4	$7{,}211\times10^{-9}$	0,003	$6{,}183\times10^{-6}$	0,0002	0,0006	8%
5	1255,583	7766,667	2921,692	3663,022	1950,825	0%

Tabela V: Resultados para o SSO.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$1{,}012\times10^{-6}$	$4{,}128\times10^{-5}$	$1{,}029\times10^{-5}$	$1{,}23\times10^{-5}$	$9{,}641\times10^{-6}$	0%
2	0,0008	0,043	0,006	0,01	0,012	0%
3	18951,163	7848043,903	248418,17	1336400,993	2393426,766	0%
4	0,007	1105,157	0,951	157,747	270,304	0%
5	0,47	14961,614	46,894	956,083	2982,338	0%

Tabela VI: Resultados para o ABC.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$7{,}344\times10^{-10}$	$9{,}976\times10^{-9}$	$6{,}121\times10^{-9}$	$6{,}027\times10^{-9}$	$3{,}036\times10^{-9}$	100%
2	4,742	60,077	16,333	20,594	14,790	0%
3	792906,864	6022824,420	2599561,270	2822379,616	1441858,856	0%
4	952,846	12749,242	6478,699	6764,657	3031,641	0%
5	72,1531	3928,588	1941,423	1869,629	1122,613	0%

Tabela VII: Resultados para o ACO.

Função	Melhor	Pior	Mediana	Média	Desvio-padrão	Taxa de Sucesso
1	$5{,}112\times10^{-9}$	$9{,}914\times10^{-9}$	$8{,}633\times10^{-9}$	$8{,}205\times10^{-9}$	$1{,}496\times10^{-9}$	100%
2	$6{,}729\times10^{-9}$	$9{,}921\times10^{-9}$	$8{,}669\times10^{-9}$	$8{,}633\times10^{-9}$	$9{,}319\times10^{-10}$	100%
3	119958,328	972501,069	418191,686	412961,230	201477,680	0%
4	$4{,}145\times10^{-9}$	$9{,}919\times10^{-9}$	$8{,}660\times10^{-9}$	$8{,}046\times10^{-9}$	$1{,}774\times10^{-9}$	100%
5	$6{,}021\times10^{-9}$	$9{,}986\times10^{-9}$	$9{,}415\times10^{-9}$	$8{,}781\times10^{-9}$	$1{,}199\times10^{-9}$	100%

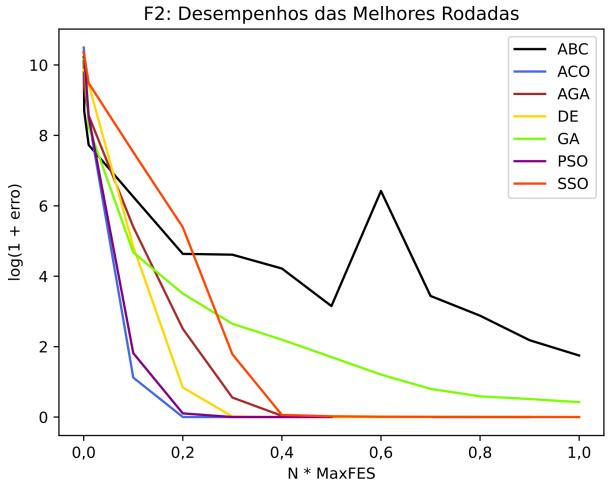

Figura 5: Gráficos de convergência para a F2 - melhor rodada.

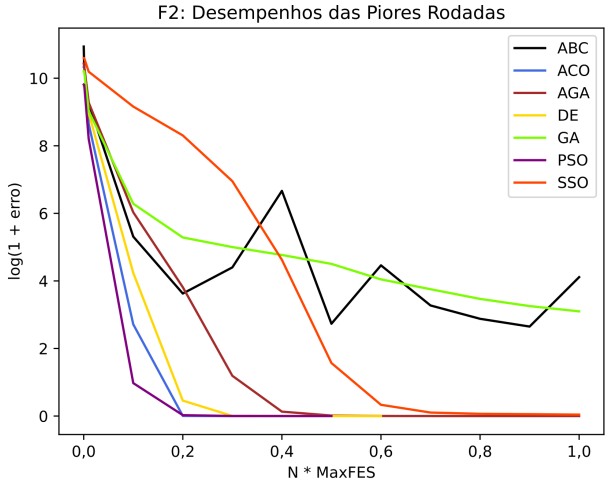

Figura 6: Gráficos de convergência para a F2 - pior rodada.

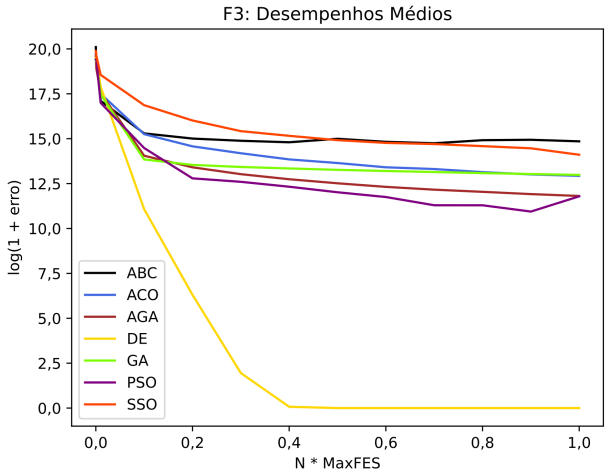

Figura 7: Gráficos de convergência para a F3 - média das soluções.

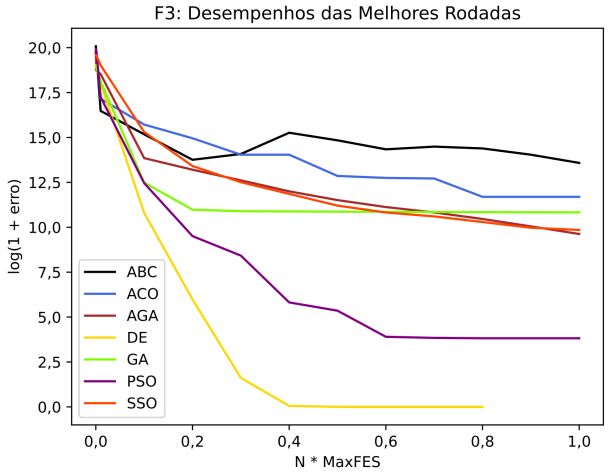

Figura 8: Gráficos de convergência para a F3 - melhor rodada.

para melhoria de parâmetros, execução simultânea ou uso de componentes de mais de um algoritmo, que sejam adaptáveis para diferentes tipos de problemas.

Referências

[1] K. F. Man, K. S. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.

[2] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*, ser. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. [Online]. Disponível em: http://link.springer.com/10.1007/978-3-662-44874-8

[3] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundations of Genetic Algorithms*, ser. Foundations of Genetic Algorithms, L. D. WHITLEY, Ed. Elsevier, 1993, vol. 2, pp. 187 – 202. [Online]. Disponível em: http://www.sciencedirect.com/science/article/pii/B9780080948324500180

[4] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, p. 19, Dec. 1997.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948 vol.4.

[6] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[7] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, 1998, pp. 69–73.

[8] E. Cuevas, M. Cienfuegos, D. Zaldívar, and M. Pérez-Cisneros, "A swarm optimization algorithm inspired in the behavior of the social-spider," *Expert Systems with Applications*, vol. 40, no. 16, pp. 6374–6384, Nov. 2013. [Online]. Disponível em: https://linkinghub.elsevier.com/retrieve/pii/S0957417413003394

[9] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[10] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155 – 1173, 2008. [Online]. Disponível em: http://www.sciencedirect.com/science/article/pii/S0377221706006333

[11] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," *KanGAL report*, vol. 2005005, no. 2005, p. 2005, 2005.

[12] L. C. Gomes, "Repositório do GitHub com implementações,"

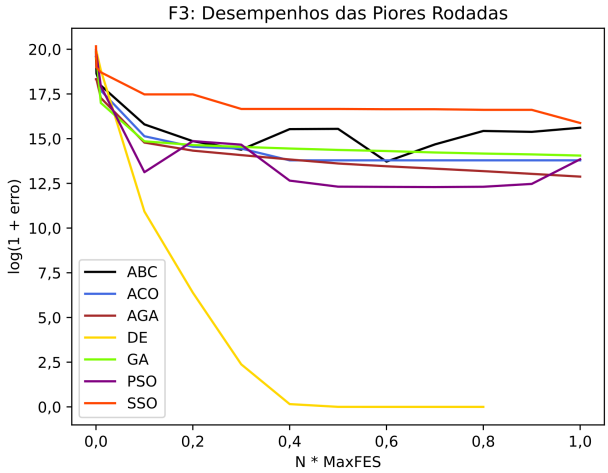

Figura 9: Gráficos de convergência para a F3 - pior rodada.

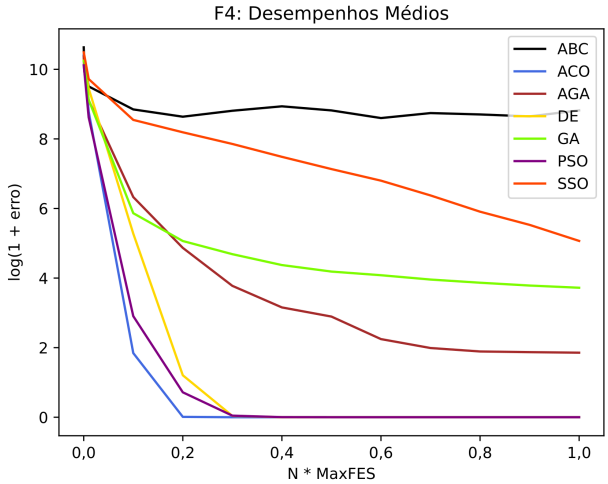

Figura 10: Gráficos de convergência para a F4 - média das soluções.

acessado em 8 de dezembro de 2020. [Online]. Disponível em: https://www.github.com/Lucas-CG/Evolution

[13] S. Wessing, "Optproblems – PyPi," acessado em 8 de dezembro de 2020. [Online]. Disponível em: https://pypi.org/project/optproblems/

[14] A. Carlisle and G. Dozier, "An off-the-shelf pso [c/cd]," in *Workshop Particle Swarm Optimization, Indianapolis*, 2001.

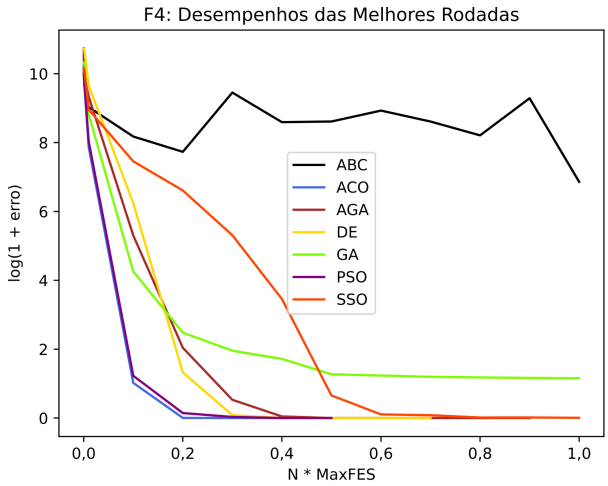

Figura 11: Gráficos de convergência para a F4 - melhor rodada.

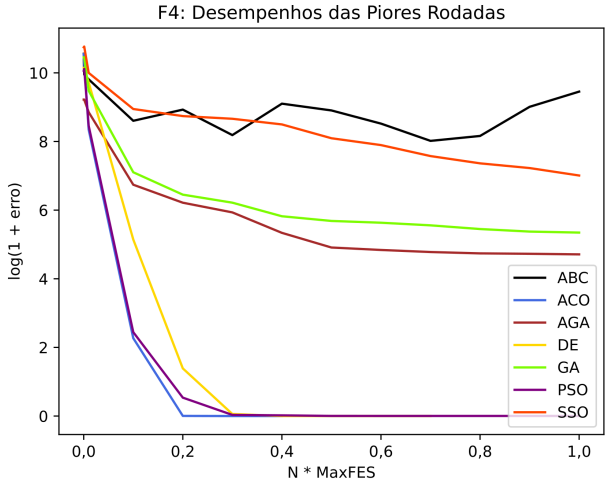

Figura 12: Gráficos de convergência para a F4 - pior rodada.

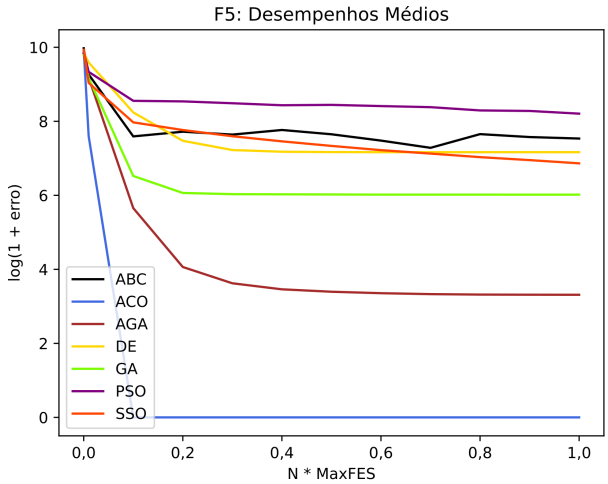

Figura 13: Gráficos de convergência para a F5 - média das soluções.

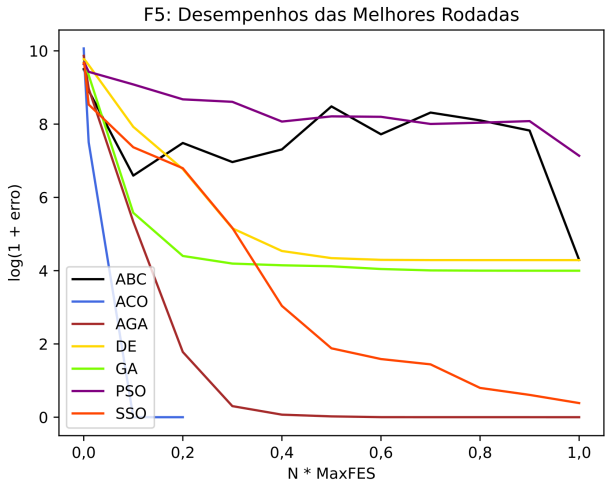

Figura 14: Gráficos de convergência para a F5 - melhor rodada.

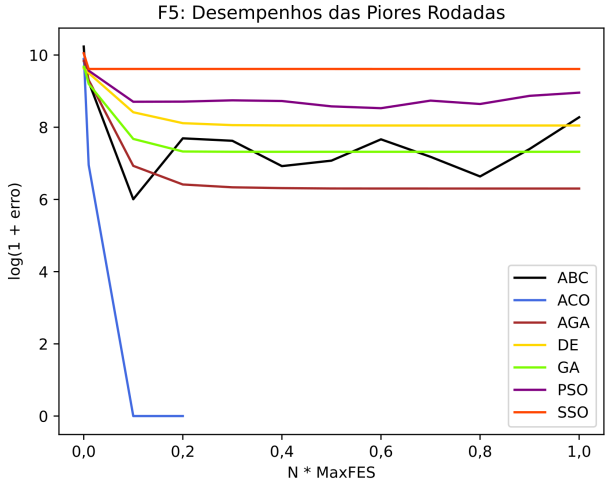

Figura 15: Gráficos de convergência para a F5 - pior rodada.