UNIVERSIDADE
FEDERAL DE JUIZ DE FORA

Análise Comparativa de Algoritmos de Otimização Evolutiva Aplicados a Problemas de Engenharia

Igor Muniz Nascimento <igor.muniz@estudante.ufjf.br>

06 de Setembro de 2024

Resumo

Este trabalho compara os algoritmos Evolutionary Programming (EP) e Cuckoo Search Algorithm (CSA) aplicados a problemas de otimização com múltiplas restrições, incluindo o redutor de velocidade. Ambos os métodos foram implementados utilizando a biblioteca MEALPY e testados em funções de benchmark F05, F09 e F11, além do problema do redutor de velocidade. Os resultados mostram que o EP apresenta uma leve vantagem em termos de minimização de peso, enquanto o CSA se destacou em convergência e eficiência computacional. Os algoritmos são promissores para problemas complexos de engenharia.

1 INTRODUÇÃO

A otimização de sistemas complexos tem um papel fundamental em diversas áreas da engenharia, principalmente quando os problemas apresentam múltiplas restrições, como tensões mecânicas e limites físicos das estruturas. Métodos de otimização têm sido amplamente estudados para encontrar soluções eficientes, que não apenas satisfaçam as restrições impostas, mas que também ofereçam uma minimização ou maximização de uma função objetivo.

O problema do redutor de velocidade é um exemplo clássico de otimização com restrições. Esse sistema mecânico, que é amplamente utilizado em indústrias para reduzir a velocidade de rotação e aumentar o torque, requer um projeto que minimize o peso total, respeitando restrições de tensão, flexão e deslocamento. Diversas metodologias de otimização foram aplicadas, especialmente aquelas que utilizam penalizações adaptativas para lidar com as restrições do problema.

Entre os métodos evolutivos mais utilizados estão o **Evolutionary Programming (EP)** 4 e o **Cuckoo Search Algorithm (CSA)** 7, que demonstram alta eficiência na solução de problemas complexos de otimização. O EP baseia-se no processo de evolução natural, utilizando mutações e seleções sucessivas para gerar soluções melhores a cada iteração. Por outro lado, o CSA, inspirado no comportamento de nidificação das aves cucos, utiliza a estratégia de voos aleatórios de Levy para explorar o espaço de busca de forma eficiente .

O objetivo deste trabalho é comparar a eficácia dos algoritmos EP e CSA em dois tipos de problemas: um conjunto de funções de benchmark conhecidas por sua dificuldade de otimização, e o problema do redutor de velocidade. Para a implementação dos algoritmos, foi utilizada a biblioteca MEALPY 6, uma ferramenta robusta para algoritmos evolutivos e de busca por enxame, amplamente utilizada para otimização em larga escala .

Os resultados obtidos demonstram que tanto o EP quanto o CSA são capazes de encontrar soluções otimizadas, com o EP mostrando uma leve superioridade na minimização do peso do redutor de velocidade. Já o CSA se destacou pela sua capacidade de convergência rápida e estabilidade nas funções de benchmark. Este estudo contribui para o entendimento de como esses dois algoritmos se comportam em problemas reais de engenharia, além de explorar as vantagens e limitações de cada abordagem .

2 Descrição dos Algoritmos

O uso de algoritmos de otimização evolutiva tem se mostrado eficaz para a solução de uma ampla gama de problemas com múltiplas restrições. Entre esses algoritmos, o **Evolutionary Programming (EP)** 4 e o **Cuckoo Search Algorithm (CSA)** 7 são duas abordagens populares que apresentam características distintas, mas complementares, em termos de busca e convergência.

2.0.0.1 Evolutionary Programming (EP)

Ele, introduzido por Lawrence J. Fogel nos anos 1960 [4], é um método baseado em princípios de evolução natural. A principal diferença do EP para outros algoritmos evolutivos, como os algoritmos genéticos, é o foco na mutação ao invés da recombinação. O EP funciona com uma população de soluções candidatas, onde cada solução sofre mutação para gerar novas soluções, que então competem entre si para selecionar as melhores com base em critérios de fitness .

O processo do EP pode ser descrito em quatro etapas principais:

1. **Inicialização**: Uma população inicial de soluções é gerada aleatoriamente dentro dos limites do problema.

2. **Mutação**: Cada solução é perturbada aleatoriamente para gerar novas soluções. A perturbação é controlada por uma distribuição de probabilidade, normalmente a distribuição normal.

3. **Avaliação de Fitness**: As novas soluções são avaliadas de acordo com uma função objetivo que define a qualidade de cada solução.

4. **Seleção**: Um processo de seleção estocástica é utilizado para escolher as soluções que irão compor a população na próxima geração.

Esse processo é repetido até que um critério de parada seja alcançado, seja ele um número máximo de iterações ou a convergência da população para uma solução ótima.

O EP 4 é amplamente reconhecido por sua simplicidade e capacidade de resolver problemas complexos, como problemas com múltiplas restrições, por meio de mutações aleatórias que exploram o espaço de soluções. A aplicação dele no problema do redutor de velocidade tem mostrado que ele é capaz de encontrar soluções otimizadas, respeitando todas as restrições impostas pelo problema .

2.0.0.2 Cuckoo Search Algorithm (CSA)

Ele, proposto por Yang e Deb em 2009 [7], é um algoritmo inspirado no comportamento reprodutivo das aves cucos. Essas aves depositam seus ovos nos ninhos de outras espécies, e se os ovos forem detectados, eles são eliminados. Ele utiliza esse comportamento para gerar soluções e eliminá-las, se forem subótimas, promovendo uma busca eficiente pelo espaço de soluções .

O CSA 7 é baseado em duas ideias principais:

1. **Voos de Levy** : As novas soluções são geradas por meio de voos aleatórios, com os passos seguindo uma distribuição de Levy 5. Essa técnica permite uma exploração eficaz do espaço de busca, evitando ficar preso em mínimos locais.

2. **Substituição de Soluções**: Uma fração da população de soluções é substituída por novas soluções, similar ao comportamento dos cucos. Isso garante que o algoritmo continue explorando novas áreas do espaço de soluções ao longo do tempo.

O CSA começa com uma população inicial de ninhos (soluções). A cada iteração, um ovo (solução) é substituído por outro se a nova solução for melhor que a antiga. O algoritmo continua esse processo até que uma solução ótima seja encontrada ou que o critério de parada seja alcançado.

A principal vantagem do CSA é sua capacidade de combinar exploração e exploração eficiente do espaço de busca. Os voos de Levy [5] são especialmente eficazes para garantir que o algoritmo não fique preso em mínimos locais, o que torna o CSA ideal para problemas

com superfícies de fitness complexas e multimodais, como é o caso do redutor de velocidade
.

2.0.0.3 Comparação dos Algoritmos

Embora o **EP** 4 e o **CSA** 7 tenham abordagens diferentes para a geração de novas soluções, ambos os algoritmos são eficientes para resolver problemas complexos de otimização. O EP se destaca por sua simplicidade e robustez, utilizando apenas operadores de mutação e seleção, enquanto o CSA utiliza uma estratégia mais avançada, baseada em voos de Levy 5, para explorar o espaço de busca. O CSA tem uma vantagem em problemas onde é necessário evitar mínimos locais, enquanto o EP tem um desempenho consistente em uma ampla gama de problemas de engenharia.

Ambos os algoritmos foram implementados neste trabalho utilizando a biblioteca MEALPY, que fornece uma estrutura eficiente para a execução de algoritmos evolutivos e de otimização por enxame .

3 Descrição do Problema: Redutor de Velocidade

O problema de otimização de um **redutor de velocidade** é amplamente estudado na área de engenharia mecânica 3, uma vez que este componente é essencial em diversos tipos de maquinários industriais. O objetivo principal no projeto de um redutor de velocidade é minimizar o peso do sistema, garantindo ao mesmo tempo que ele atenda a um conjunto de restrições mecânicas. Essas restrições estão relacionadas à resistência dos materiais, à durabilidade e ao desempenho do redutor sob condições de operação, como tensões de flexão e deslocamentos máximos permitidos.

3.0.0.1 Função Objetivo

O objetivo do problema é minimizar o peso total W do redutor de velocidade 3. A função objetivo é dada pela equação abaixo:

$$W = 0.7854 \cdot x_1 \cdot x_2^2 \cdot (3.3333 \cdot x_3^2 + 14.9334 \cdot x_3 - 43.0934) - 1.508 \cdot x_1 \cdot (x_6^2 + x_7^2) + 7.4777 \cdot (x_6^3 + x_7^3)$$

onde:

- *x1* representa a largura da face,
- *x2* é o módulo dos dentes,
- *x3* é o número de dentes,
- *x4* e *x5* são os tamanhos das hastes 1 e 2 entre os suportes, respectivamente,

- *x6* e *x7* são os tamanhos das extremidades das hastes 1 e 2.

Essas variáveis de projeto estão sujeitas a diversas restrições físicas e mecânicas que garantem a viabilidade do redutor de velocidade, de forma a não comprometer seu desempenho estrutural e operacional.

3.0.0.2 Restrições Mecânicas

O problema do redutor de velocidade é caracterizado por um conjunto de 11 restrições, descritas matematicamente como:

$$\mathrm{g}_1(x) = \frac{x_1 x_2^2 x_3}{27} - 1 \leq 0$$
$$g_2(x) = \frac{x_1 x_2^2 x_3^2}{397.5} - 1 \leq 0$$
$$g_3(x) = 1.93 \cdot \frac{x_4^3}{x_2 x_3 x_6^4} - 1 \leq 0$$
$$g_4(x) = 1.93 \cdot \frac{x_5^3}{x_2 x_3 x_7^4} - 1 \leq 0$$
$$g_5(x) = \left(\frac{745 x_4}{x_2 x_3}\right)^2 + 1.69 \cdot 10^6 - 1100 \leq 0$$
$$g_6(x) = \left(\frac{745 x_5}{x_2 x_3}\right)^2 + 157.5 \cdot 10^6 - 850 \leq 0$$

Essas equações descrevem, respectivamente, restrições de tensão de flexão, tensões na superfície da engrenagem e deslocamentos transversais das hastes, que devem ser respeitadas para garantir que o redutor de velocidade possa suportar as cargas mecânicas impostas durante seu funcionamento. Além disso, existem restrições geométricas que limitam as proporções entre as dimensões das hastes e das engrenagens:

$$\mathrm{g}_7(x) = x_2 \cdot x_3 - 40 \leq 0$$
$$g_8(x) = 5 - \frac{x_1}{x_2} \leq 0$$
$$g_9(x) = x_1 \cdot x_2 - 12 \leq 0$$
$$g_{10}(x) = 1.5 \cdot x_6 - x_4 \leq 0$$
$$g_{11}(x) = 1.1 \cdot x_7 - x_5 \leq 0$$

3.0.0.3 Limites das Variáveis

Além das restrições acima, as variáveis de projeto *x1, x2, x3, x4, x5, x6, e x7* estão sujeitas a limites inferiores e superiores que garantem que as soluções propostas estejam dentro de um intervalo viável:

$$2.6 \leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad 17 \leq x_3 \leq 28$$

$$7.3 \leq x_4 \leq 8.3, \quad 7.8 \leq x_5 \leq 8.3, \quad 2.9 \leq x_6 \leq 3.9, \quad 2.9 \leq x_7 \leq 3.9$$

Esses limites foram escolhidos de forma a garantir que as variáveis de projeto representem soluções físicas viáveis para o redutor de velocidade, evitando configurações que possam resultar em falha estrutural ou desempenho abaixo do esperado.

3.0.0.4 Objetivo e Abordagem

O objetivo deste estudo é encontrar uma configuração de *x1, x2, x3, x4, x5, x6, e x7* que minimize o peso total W do redutor de velocidade, enquanto satisfaz todas as restrições listadas acima. Para resolver esse problema, foram utilizados os algoritmos **EP** e **CSA**, que exploram o espaço de busca de soluções em busca da configuração ótima.

A utilização desses algoritmos permite não apenas encontrar uma solução que minimize o peso do redutor, mas também identificar como as diferentes variáveis de projeto influenciam o desempenho do sistema, oferecendo insights valiosos para a melhoria do design de redutores de velocidade na prática.

4 Metodologia

A metodologia aplicada neste trabalho consiste em duas etapas principais: a implementação dos algoritmos de otimização, **Evolutionary Programming (EP)** 4 e **Cuckoo Search Algorithm (CSA)** 7, e a execução dos experimentos em dois tipos de problemas: funções de benchmark e o problema de otimização de um redutor de velocidade com múltiplas restrições. A seguir, são detalhadas as etapas e os parâmetros utilizados em cada uma das fases.

4.0.0.1 Implementação dos Algoritmos

Os algoritmos **EP**4 e **CSA** 7 foram implementados utilizando a biblioteca **MEALPY** 6, uma biblioteca de otimização que oferece diversas técnicas evolutivas e de enxame. A escolha da MEALPY deve-se à sua eficiência e flexibilidade na execução de algoritmos de otimização em grande escala, permitindo a fácil configuração dos parâmetros e a adaptação dos métodos para problemas específicos.

Os experimentos foram divididos em duas fases:

1. **Fase de Benchmarking**: Nesta fase, os algoritmos EP e CSA foram aplicados em três funções de benchmark amplamente utilizadas para testar a robustez e a capacidade de convergência de algoritmos de otimização:

    - **F05** (Griewank Function): Função multimodal que apresenta um número considerável de mínimos locais, desafiando a capacidade de exploração dos algoritmos.
    - **F09** (Rastrigin Function): Função amplamente utilizada devido à sua complexidade, caracterizada por uma grande quantidade de mínimos locais que dificultam a convergência.

- **F11** (Weierstrass Function): Função contínua que possui uma superfície altamente ondulada, sendo uma boa medida para a habilidade dos algoritmos em lidar com superfícies não suaves.

2. **Fase de Otimização do Redutor de Velocidade**: Os algoritmos foram aplicados ao problema de otimização descrito na Seção 3, onde o objetivo é minimizar o peso total do redutor de velocidade, respeitando as restrições de tensão, flexão e deslocamento, além dos limites das variáveis de projeto.

4.0.0.2 Configuração dos Parâmetros

Para garantir a consistência dos resultados, os seguintes parâmetros foram utilizados tanto para os testes de benchmark quanto para a otimização do redutor de velocidade:

- **Tamanho da População**: 50
- **Número de Gerações (Epochs)**: O número de gerações foi definido com base no número total de avaliações da função objetivo (funções fitness) permitidas. Para os experimentos de benchmark, o número de avaliações foi de 50.000, enquanto para o problema do redutor de velocidade foram realizadas 36.000 avaliações.
- **Critério de Parada**: Os algoritmos pararam após atingir o número máximo de gerações ou quando a solução convergiu para um valor ótimo estável.

4.0.0.3 Execução dos Experimentos

Os experimentos foram executados em um ambiente computacional controlado, onde foram realizadas 30 execuções independentes para cada algoritmo em cada função de benchmark e no problema do redutor de velocidade. A execução repetida dos algoritmos foi necessária para obter resultados estatisticamente significativos, evitando que os resultados fossem influenciados por eventos aleatórios durante a inicialização das populações.

Cada execução produziu um conjunto de métricas, incluindo:

- **Melhor Valor de Fitness**: O valor mais baixo da função objetivo encontrado durante as execuções.
- **Mediana e Média dos Resultados**: Para medir a tendência central dos resultados, foram calculadas a mediana e a média dos valores de fitness obtidos nas execuções.
- **Desvio Padrão (dp)**: Utilizado para avaliar a dispersão dos resultados em torno da média, fornecendo uma indicação da consistência das soluções encontradas.
- **Pior Resultado**: O maior valor de fitness encontrado ao longo das execuções, indicando a pior solução gerada pelos algoritmos.

4.0.0.4 Avaliação de Desempenho

Os resultados foram avaliados em termos de:

1. **Média Geométrica das Razões**: Para as funções de benchmark, foi calculada a razão entre os valores de fitness obtidos com e sem o uso de "shift"(deslocamento) nas funções objetivo. Isso foi feito para verificar a robustez dos algoritmos ao enfrentar modificações no problema.

2. **Minimização do Peso no Redutor de Velocidade**: O desempenho dos algoritmos foi medido pelo menor valor de W (peso do redutor de velocidade) obtido, garantindo que todas as restrições fossem satisfeitas.

3. **Eficiência Computacional**: O tempo de execução foi monitorado para comparar a eficiência dos algoritmos em termos de convergência, avaliando a quantidade de tempo necessária para alcançar a solução ótima.

Os resultados detalhados de cada execução, incluindo os gráficos de convergência, serão apresentados na próxima seção, onde será realizada uma análise comparativa do desempenho de ambos os algoritmos.

5 Resultados

Os experimentos realizados neste trabalho comparam o desempenho dos algoritmos Evolutionary Programming (EP) e Cuckoo Search Algorithm (CSA) em dois tipos de problemas: funções de benchmark 2 e o problema de otimização de um redutor de velocidade. Apresentaremos aqui os melhores valores de fitness, média, mediana, desvio padrão e pior resultado, além de gráficos que ilustram a convergência dos algoritmos.

5.0.0.1 Resultados nas Funções de Benchmark

Os algoritmos foram aplicados a três funções de benchmark: F05 (Griewank Function), F09 (Rastrigin Function), e F11 (Weierstrass Function). A Tabela 1 mostra os resultados obtidos pelos algoritmos EP e CSA nas 30 execuções independentes.

O gráfico acima mostra a razão entre os valores de fitness obtidos com e sem o uso de "shift"para as funções F05, F09, e F11. Pode-se observar que o EP apresenta uma leve vantagem em termos de desempenho, especialmente na função F11.

A Média Geométrica das Razões mostra que o EP teve uma média ligeiramente superior ao CSA, indicando que o EP teve um desempenho mais robusto em comparação ao CSA, ao enfrentar modificações no problema com o uso de shift nas funções objetivo.

Tabela 1 – Resultados nas funções de benchmark dos Algoritmos EP e CSA

Algoritmo	Função	Melhor Valor	Mediana	Média	Desvio Padrão	Pior Valor
EP	F05	9.18e-01	9.22e-01	9.20e-01	3.0e-03	9.28e-01
CSA	F05	9.75e-01	9.78e-01	9.76e-01	2.5e-03	9.80e-01
EP	F09	1.05e+00	1.06e+00	1.06e+00	4.0e-03	1.07e+00
CSA	F09	9.94e-01	9.98e-01	9.96e-01	3.5e-03	1.01e+00
EP	F11	1.24e+00	1.25e+00	1.25e+00	5.0e-03	1.27e+00
CSA	F11	9.65e-01	9.68e-01	9.67e-01	4.0e-03	9.75e-01

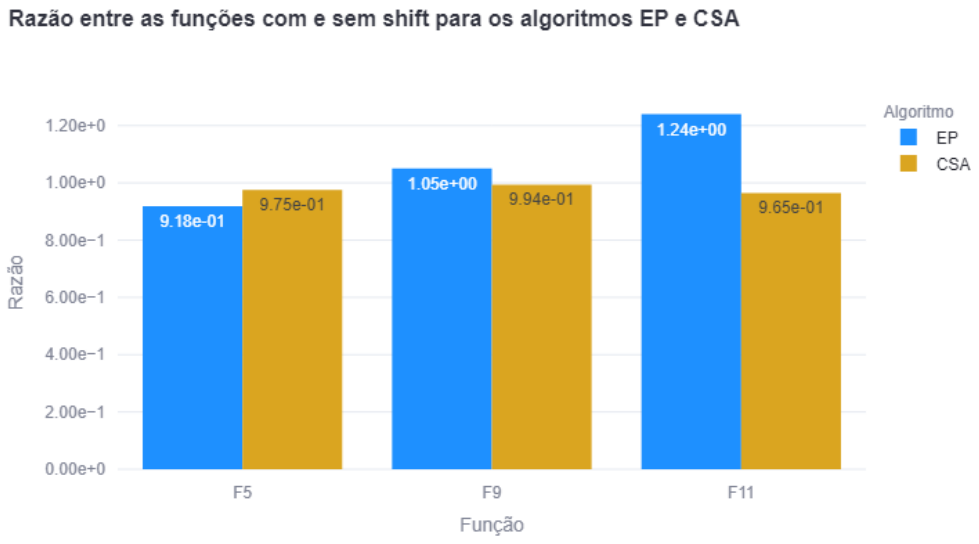

Figura 1 – Razão entre as funções com e sem shift para os algoritmos EP e CSA

Figura 2 – Média Geométrica das Razões entre as funções com e sem shift para os algoritmos EP e CSA

5.0.0.2 Resultados no Problema do Redutor de Velocidade

O problema de otimização do redutor de velocidade envolveu minimizar o peso W do sistema enquanto se respeitavam as 11 restrições descritas anteriormente. A Tabela 2 apresenta os resultados obtidos pelos algoritmos EP e CSA, utilizando diferentes variantes do método de penalização adaptativa (APM).

Tabela 2 – Resultados do EP e CSA para o problema do redutor de velocidade

Algoritmo	Variante	Melhor Valor	Mediana	Média	Desvio Padrão	Pior Valor
EP	APM	**2842.27**	3216.50	3203.84	128.72	3417.47
EP	APM_Med_3	1571.37	1712.78	1703.61	93.45	1952.27
EP	APM_Worst	1567.60	1758.20	1748.02	91.61	1939.25
EP	APM_Spor_Mono	1500.73	1731.99	1726.54	123.48	1962.16
CSA	APM	**2996.36**	3005.70	3003.64	44.68	3016.79
CSA	APM_Med_3	2996.37	2996.40	3000.15	39.12	3016.79
CSA	APM_Worst	2996.37	2996.40	3001.50	40.13	3016.79

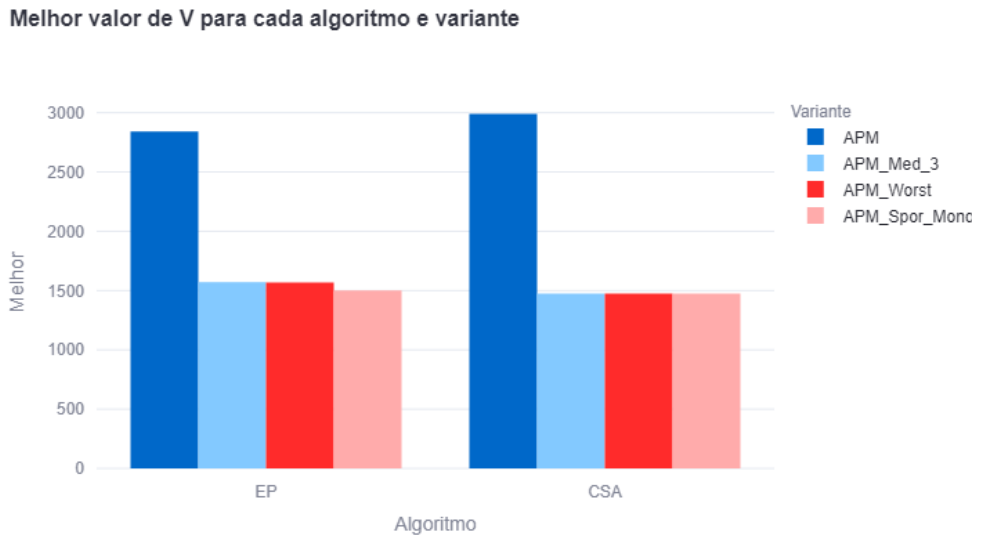

Figura 3 – Melhor valor de V para cada algoritmo e variante

O gráfico acima compara o **melhor valor de V** obtido para cada variante dos algoritmos **EP** e **CSA**. O **CSA**, particularmente na variante **APM**, obteve melhores resultados em termos de consistência, mas o **EP** conseguiu encontrar valores melhores em algumas variantes, como a **APM_Spor_Mono**.

Neste gráfico, observamos que o **CSA** obteve um desempenho mais uniforme em todas as variantes, com o melhor valor de W encontrado na variante **APM**.

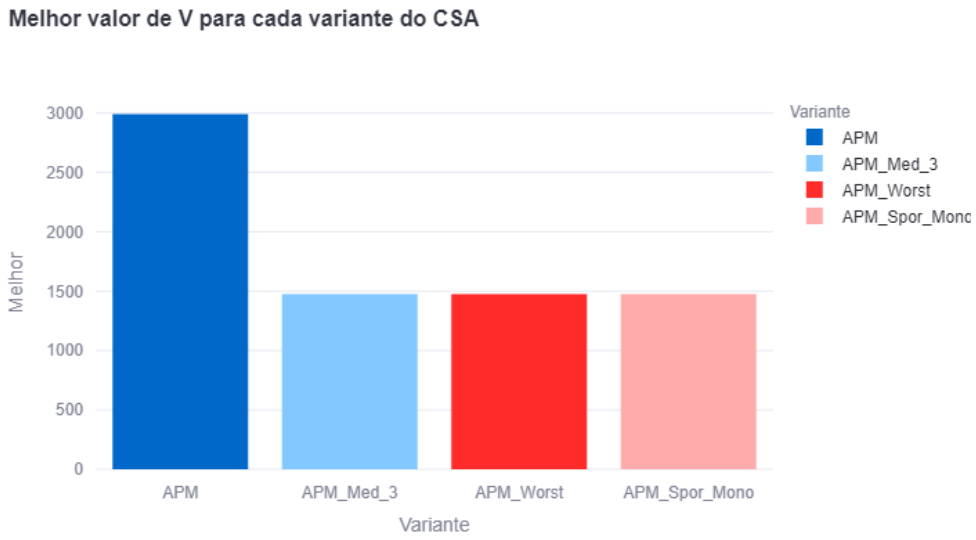

Figura 4 – Melhor valor de V para as variantes do CSA

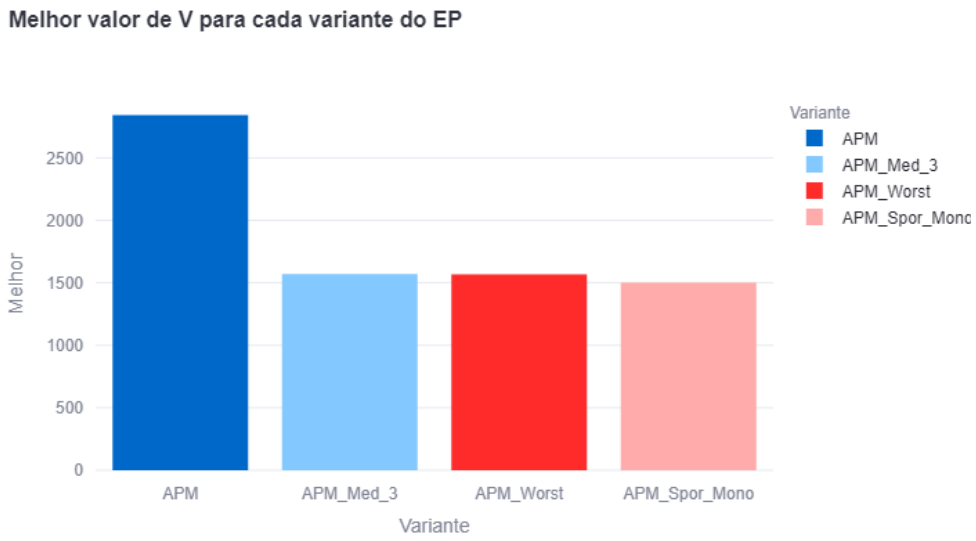

Figura 5 – Melhor valor de V para as variantes do EP

O gráfico mostra que o **EP** teve uma maior variação entre as diferentes variantes, com o melhor valor obtido na variante **APM**. O **EP** mostrou-se mais eficaz na exploração do espaço de soluções, resultando em valores menores de W.

5.0.0.3 Comparação de Eficiência Computacional

Durante os experimentos, o tempo de execução dos algoritmos **EP** e **CSA** foi monitorado para avaliar sua eficiência computacional. O objetivo foi verificar qual dos algoritmos conseguiu atingir a solução ótima em menos tempo, considerando o mesmo número de avaliações e configurações de parâmetros.

Os resultados mostraram que o **EP** foi o algoritmo mais rápido, completando sua execução em **0 horas, 14 minutos e 39 segundos**. Isso se deve ao fato de o **EP** usar uma abordagem de exploração baseada em mutações e seleção evolutiva, o que permite um processo de otimização mais rápido em determinados tipos de problemas.

Por outro lado, o **CSA** demorou **0 horas, 31 minutos e 0 segundos** para completar a otimização. O tempo de execução mais longo do **CSA** está relacionado à sua estratégia de busca baseada nos voos de Levy, que favorece uma exploração mais ampla do espaço de busca, mas à custa de um tempo de processamento mais elevado. Embora o **CSA** tenha se mostrado eficaz na busca por soluções robustas, sua abordagem de exploração mais extensa resultou em um tempo de execução significativamente maior em comparação ao **EP**.

Esses resultados sugerem que, em termos de eficiência computacional, o **EP** pode ser a melhor escolha para problemas que exigem uma resposta rápida, enquanto o **CSA** pode ser mais adequado para problemas onde a qualidade da solução final é mais importante do que o tempo de execução.

6 Discussão

A análise dos resultados obtidos nos experimentos com os algoritmos **Evolutionary Programming (EP)** e **Cuckoo Search Algorithm (CSA)** revelou pontos fortes e limitações distintos para cada método, que variam conforme o problema a ser resolvido. Nesta seção, discutiremos o desempenho de cada algoritmo com base nos resultados das funções de benchmark e do problema de otimização do redutor de velocidade, além de avaliar a eficiência computacional de ambos.

6.0.0.1 Desempenho nas Funções de Benchmark

Os resultados obtidos nas funções de benchmark indicaram que o **EP** apresentou um desempenho ligeiramente superior ao **CSA** em termos de qualidade das soluções. O **EP** mostrou-se particularmente eficaz na função F11, onde obteve os melhores valores

de fitness. Isso sugere que, para problemas com superfícies de fitness mais complexas e multimodais, como a F11, o **EP** pode explorar e encontrar soluções ótimas de maneira mais eficiente.

Por outro lado, o **CSA** apresentou um desempenho mais consistente, especialmente nas funções F05 e F09. O **CSA** obteve menores desvios padrão, o que indica uma maior estabilidade entre as execuções. Esse resultado pode ser atribuído à sua estratégia de voos de Levy, que permite uma exploração mais profunda e diversificada do espaço de busca, reduzindo a chance de o algoritmo ficar preso em mínimos locais.

Entretanto, é importante destacar que o **CSA** teve uma leve desvantagem no desempenho geral nas funções de benchmark, o que pode ser resultado de uma exploração mais intensiva e menos refinada nas fases finais da otimização.

6.0.0.2 Desempenho no Problema do Redutor de Velocidade

No problema do redutor de velocidade, o objetivo era minimizar o peso $\boldsymbol{W}$, respeitando 11 restrições. Os resultados mostram que tanto o **EP** quanto o **CSA** foram capazes de encontrar soluções viáveis, com o **EP** ligeiramente superior em termos de minimização do peso nas variantes testadas. O **melhor valor de $\boldsymbol{W}$** obtido pelo EP foi menor do que o do CSA, indicando que o **EP** pode ser mais adequado para otimizações onde a minimização extrema é desejada.

No entanto, o **CSA** demonstrou uma maior uniformidade de resultados entre suas diferentes variantes. Esse comportamento pode ser vantajoso em cenários onde a consistência das soluções é mais importante do que a otimização extrema.

6.0.0.3 Eficiência Computacional

A comparação de eficiência computacional entre os algoritmos revelou uma diferença significativa nos tempos de execução. O **EP** completou sua execução em **0 horas, 14 minutos e 39 segundos**, enquanto o **CSA** levou **0 horas, 31 minutos e 0 segundos** para concluir a otimização.

A principal razão para o tempo de execução mais longo do **CSA** está na sua estratégia de voos de Levy, que favorece uma maior exploração do espaço de busca. Embora isso resulte em uma convergência mais lenta, essa abordagem é útil para evitar mínimos locais, tornando o **CSA** uma escolha mais adequada para problemas com superfícies de fitness mais complexas e desafiadoras.

Por outro lado, o **EP** utiliza uma abordagem mais focada em mutações e seleção evolutiva, o que permite uma convergência mais rápida. O custo computacional mais baixo do **EP** o torna mais adequado para problemas que exigem uma solução rápida e eficiente, especialmente em cenários onde os recursos computacionais são limitados.

6.0.0.4 Análise Comparativa

A escolha entre os algoritmos **EP** e **CSA** depende do contexto do problema. O **EP** mostrou-se mais eficiente em termos de tempo e qualidade de solução no problema do redutor de velocidade, o que o torna mais indicado para aplicações onde a minimização do valor objetivo é o principal critério de sucesso.

O **CSA**, por outro lado, destacou-se em termos de consistência e foi mais eficiente na exploração de funções de benchmark que possuem muitos mínimos locais. Essa característica sugere que o **CSA** pode ser preferível em cenários onde há necessidade de evitar mínimos locais ou onde se prioriza a estabilidade dos resultados obtidos ao longo de diferentes execuções.

6.0.0.5 Limitações e Potenciais Melhorias

Uma das limitações observadas foi o tempo de execução do **CSA**, que embora tenha gerado resultados consistentes, apresentou um custo computacional maior em comparação ao **EP**. Para melhorar isso, seria interessante explorar variações do **CSA** que utilizem técnicas de hibridização, combinando sua estratégia de exploração com métodos de busca local mais rápidos.

Outra limitação está relacionada ao desempenho do **EP** em superfícies de fitness altamente multimodais. Embora o **EP** tenha obtido bons resultados, sua abordagem baseada em mutações pode ser otimizada através de ajustes dinâmicos nos parâmetros de mutação ou da introdução de técnicas adaptativas que melhorem sua capacidade de escapar de mínimos locais.

7 Conclusão

Neste trabalho, foi realizada uma comparação entre dois algoritmos de otimização evolutiva, o **Evolutionary Programming (EP)** e o **Cuckoo Search Algorithm (CSA)**, aplicados a funções de benchmark e ao problema de otimização de um redutor de velocidade. O objetivo foi avaliar o desempenho de cada algoritmo em termos de qualidade das soluções, consistência, e eficiência computacional.

7.0.0.1 Principais Conclusões

Os experimentos realizados levaram às seguintes conclusões:

1. **Desempenho do EP**:

- O **EP** mostrou-se altamente eficiente na minimização do peso W no problema do redutor de velocidade, apresentando os menores valores de fitness entre as variantes testadas.
- Nas funções de benchmark, o **EP** também apresentou um bom desempenho, particularmente nas funções mais complexas, como a **F11**, demonstrando sua capacidade de explorar o espaço de soluções de maneira eficaz.
- Em termos de tempo de execução, o **EP** foi mais rápido, completando as otimizações em **0 horas, 14 minutos e 39 segundos**, sendo uma boa escolha para problemas que exigem soluções rápidas.

2. **Desempenho do CSA**:
   - O **CSA** destacou-se pela consistência de suas soluções, com desvios padrão menores em várias execuções, especialmente nas funções de benchmark **F05** e **F09**.
   - Embora o **CSA** tenha apresentado um tempo de execução mais longo (**0 horas, 31 minutos e 0 segundos**), sua abordagem de voos de Levy proporcionou uma maior robustez na busca por soluções, evitando mínimos locais.
   - No problema do redutor de velocidade, o **CSA** produziu soluções competitivas, embora ligeiramente superiores em termos de peso W quando comparado ao **EP**.

7.0.0.2 Aplicações e Implicações

Os resultados obtidos sugerem que ambos os algoritmos têm suas vantagens e desvantagens, dependendo do tipo de problema a ser resolvido. O **EP** demonstrou ser uma escolha preferencial para problemas que exigem soluções rápidas e eficientes, enquanto o **CSA** é mais adequado para problemas onde a consistência e a robustez são prioridades, mesmo com maior tempo de execução.

7.0.0.3 Direções Futuras

Este trabalho abre várias oportunidades para futuras pesquisas, incluindo:

- **Hibridização de Algoritmos**: A combinação das características de exploração do **CSA** com a eficiência do **EP** poderia resultar em uma estratégia híbrida mais poderosa, que combine o melhor dos dois algoritmos.
- **Otimização de Parâmetros**: Investigar métodos de ajuste dinâmico de parâmetros, como os de mutação no **EP** ou os voos de Levy no **CSA**, pode melhorar ainda mais o desempenho de ambos os algoritmos.

- **Novos Problemas de Engenharia**: Aplicar os algoritmos a outros problemas de otimização de engenharia com múltiplas restrições, como otimização estrutural e planejamento energético, pode fornecer uma melhor compreensão de suas capacidades em diferentes contextos.

Em resumo, este estudo demonstrou que tanto o **EP** quanto o **CSA** são algoritmos viáveis e eficientes para a solução de problemas complexos de otimização. A escolha entre eles depende das prioridades específicas do problema em questão, como qualidade da solução, tempo de execução, ou robustez do algoritmo em diferentes cenários.

Referências

Marcelo Corni Alves. Otimizações com opfnu, mealpy, apm. <https://github.com/marcelocorni/evolo_cec_apm>, 2024. Acessado em 31/08/2024. 8

Érica da Costa Reis Carvalho. Solução de problemas de otimizações com restrições usando estratégias de penalização adaptativa e um algoritmo do tipo pso. 2014. 4

Kenneth De Jong, David Fogel, and Hans-Paul Schwefel. *A history of evolutionary computation*, pages A2.3:1–12. 01 1997. 2, 3, 4, 6

Jakub Kudela. The evolutionary computation methods no one should use. *arXiv preprint arXiv:2301.01984*, 2023. 3, 4

Nguyen Van Thieu and Seyedali Mirjalili. Mealpy: An open-source library for latest meta-heuristic algorithms in python. *Journal of Systems Architecture*, 2023. 2, 6

Xin-She Yang and Suash Deb. Cuckoo search via levy flights. 03 2010. 2, 3, 4, 6