Instituto Tecnológico de Aeronáutica
CT-213: Inteligência Artificial para Robótica Móvel
Lab 5 - Estratégias Evolutivas

Bruno Benjamim Bertucci - Turma 23.2

1 Estratégias Evolutivas

No intuito de obter métodos computacionais eficientes para a otimização de funções, foram desenvolvidos as chamadas Estratégias Evolutivas. Elas são inspiradas em conceitos de evolução da natureza, como mutação e seleção, junto com o conceito estatístico de distribuição normal.

De forma resumida, as Estratégias Evolutivas seguem a seguinte lógica: No espaço vetorial formado pelas variáveis da função a qual se quer otimizar, são distribuidos uma população fixa de pontos, criados inicialmente em uma distribuição normal seguindo um "chute inicial"de média e matriz de covariância. Os valores da função para cada um desses pontos são calculados e ordenados, e então uma dada quantidade dos melhores pontos são escolhidos, em uma analogia com a seleção natural, para calcular a média e a matriz de covariância de uma nova "geração"de pontos, de forma semelhante a uma mutação na natureza.

Dessa forma, os valores das variáveis da função "evoluem", aproximando-se cada vez mais de um resultado ótimo.

2 Estratégia Evolutiva Símples

Uma das formas de implementar uma Estratégia Evolutiva é denominada Estratégia Evolutiva Sìmples, ou *Simple Evolution Strategy (SES)*. De forma resumida, ela efetua a evolução da média da distribuição normal segundo a Equação 1, e da matriz de covariância segundo a Equação 2, onde $m^{(g)}$ e $C^{(g)}$ denotam, respectivamente, a média e a matriz de covariância na geração de número g, μ é o número dos melhores pontos escolhidos (hiperparâmetro), λ é o tamanho da população amostrada (hiperparâmetro), e $s_{i:\lambda}^{(g)}$ denota o i-ésimo melhor ponto da população de tamanho λ da geração g.

$$m^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} s_{i:\lambda}^{(g)} \tag{1}$$

$$C^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} (s_{i:\lambda}^{(g+1)} - m^{(g)})(s_{i:\lambda}^{(g+1)} - m^{(g)})^T \tag{2}$$

2.1 Implementação

A implementação da Estratégia Evolutiva Símples foi feita a partir de uma classe denominada SimpleEvolutionStrategy, com um construtor, um método ask() e um método tell().

1. O construtor recebe os hiperparâmetros λ e μ, a média e a matriz de covariância do "chute inicial". Ele armazena esses valores para serem usados nos outros métodos, e cria a amostra com distribuição normal baseada nos parâmetros do "chute inicial".

2. O método ask() simplesmente retorna essa amostra quando ele é chamado. O intuito é que ele forneça essa amostra para que os valores da função para cada ponto dela sejam calculados, e que esses valores sejam, em seguida, fornecidos para o método tell(), que deve ser chamado após a avaliação dos pontos da amostra.

3. Finalmente, o método tell(), ao receber os valores para cada ponto da amostra, ordena os pontos com base neles, seleciona os μ melhores, e efetua a evolução da matriz de covariância e da média. Finalmente, uma nova distribuição normal de pontos é gerada, com o mesmo número de pontos, mas com os novos parâmetros de matriz de covariância e média.

A cada iteração, os passos 2 e 3 são repetidos até que uma condição de parada seja atingida, no caso, o número máximo de iterações. Com essa implementação, o algorítmo SES busca otimizar funções.

3 Comparação entre SES e CMA-ES

O desempenho do algorítmo SES implementado foi testado utilizando uma variedade de funções de duas variáveis. São elas: Translated Sphere, Ackley, Schaffer function N. 2, e Rastrigin 2D (Equações 3, 4, 5, 6, respectivamente).

$$f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 2)^2 \tag{3}$$

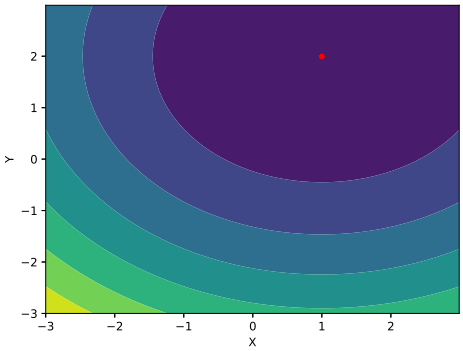
$$f(x_1, x_2) = -20e^{-0,2\sqrt{0,5(x_1^2+x_2^2)}} - e^{0,5(cos(2\pi x_1)+cos(2\pi x_2))} + e + 20 \tag{4}$$

$$f(x_1, x_2) = 0,5 + \frac{sin(x_1^2 - x_2^2)^2 - 0,5}{(1 + 0,001(x_1^2 + x_2^2))^2} \tag{5}$$

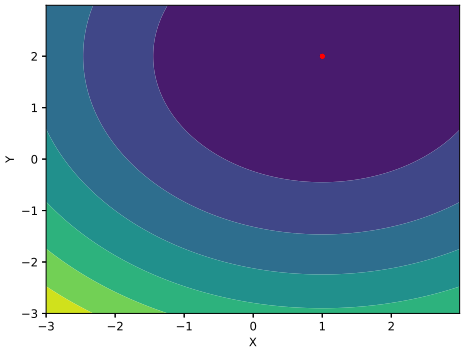
$$f(x_1, x_2) = 20 + \sum_{i=1}^{2}(x_i^2 - 10cos(2\pi x_i)) \tag{6}$$

Adicionalmente, as funções também foram otimizadas usando um algorítmo mais complexo de Estratégia de Evolução, denominado *Covariance Matrix Adaptation Evolution Strategy (CMA-ES)*, para fins de comparação. A comparação foi feita utilizando o método de *Monte Carlo*, no qual uma série de repetições da otimização foi realizada com cada algorítmo. Ao fim do teste, é calculado, para cada algorítmo, a média dos valores encontrados pelas amostras e o melhor valor encontrado para cada geração.

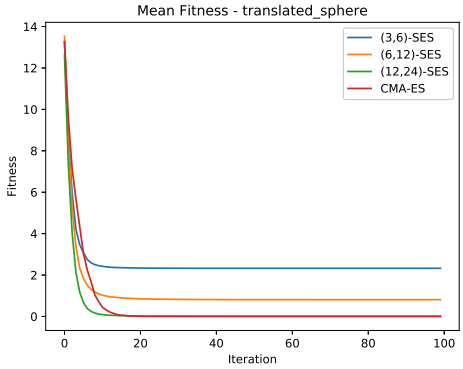
Ao todo, foram realizados, para cada algorítmo, 200 testes com 100 iterações cada. Figuras que ilustram o teste para as funções Translate Sphere, Ackley, Schaffer N. 2, e Rastrigin 2D, encontram-se nas Figuras 1, 2, 3, 4, respectivamente. Nas figuras, o algorítmo denominado, por exemplo, (3,6)-SES é aquele que considera $\mu = 3$ e $\lambda = 6$. Em todos os testes, o CMA-ES é executado com $\mu = 3$ e $\lambda = 6$.



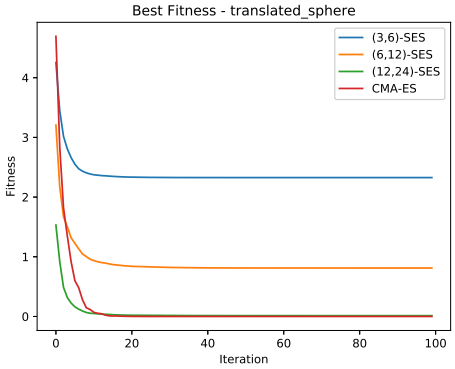
(a) Otimização com (12,24)-SES.



(b) Otimização com CMA-ES.



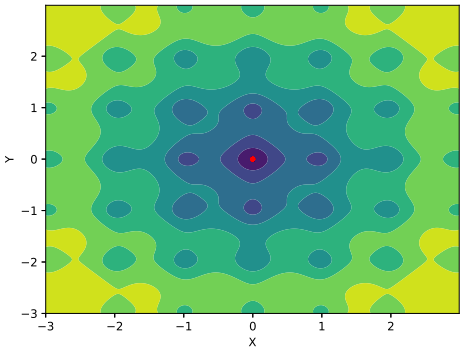
(c) Evolução do valor médio da função.



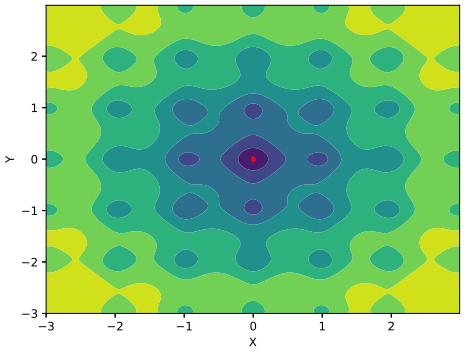
(d) Evolução do valor ótimo da função.

Figura 1: Otimização da função Translated Sphere.

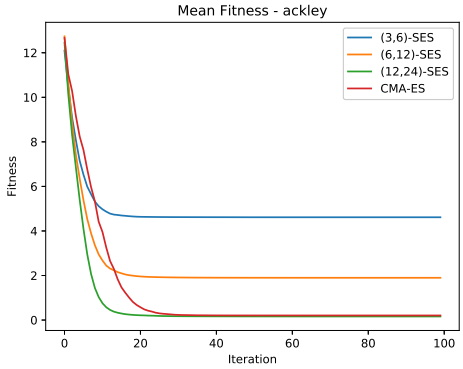
Nesse teste, observa-se que o algorítmo (12,24)-SES chegou a uma solução semelhante àquela encontrada pelo CMA-ES, sendo essas muitos próximas ao mínimo global da função. Enquanto isso, os algorítmos (3,6)-SES e (6,12)-SES convergiram para valores maiores, indicando que ficaram presos em algum mínimo local.



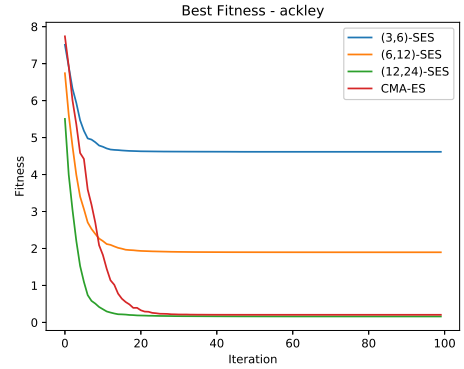
(a) Otimização com (12,24)-SES.



(b) Otimização com CMA-ES.



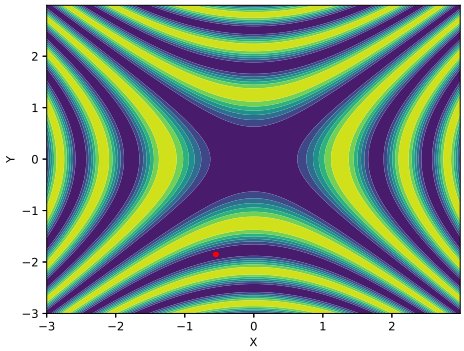
(c) Evolução do valor médio da função.



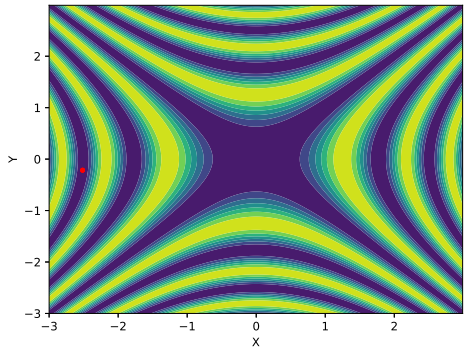
(d) Evolução do valor ótimo da função.

Figura 2: Otimização da função Ackley.

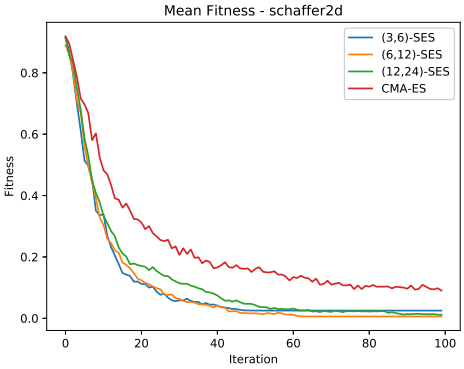
A função Ackley rendeu resultados semelhantes à Translated Sphere. Em ambos as funções, há uma tendência clara de aproximação dos respectivos mínimos globais conforme se aproxíma dos pontos destes, o que pode ser observado pelas figuras das curvas de nível das funções. Em virtude disso que alguns dos algorítmos testados chegaram em valores muito próximos do mínimo global.



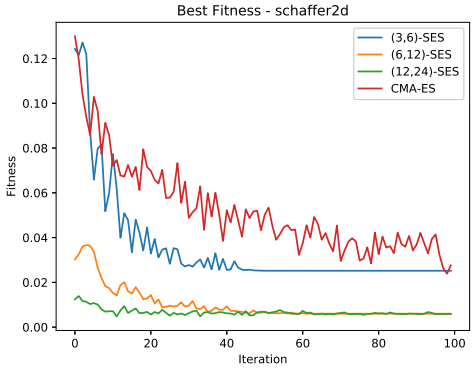
(a) Otimização com (12,24)-SES.



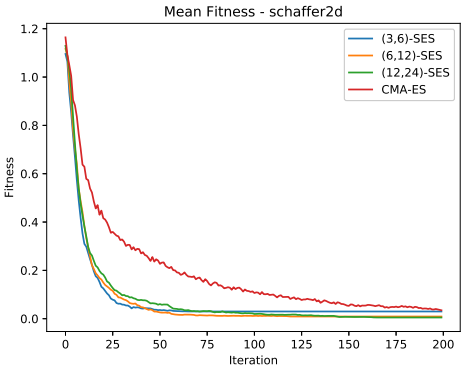
(b) Otimização com CMA-ES.



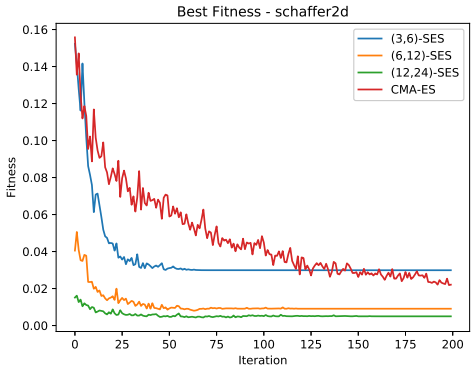
(c) Evolução do valor médio da função.



(d) Evolução do valor ótimo da função.



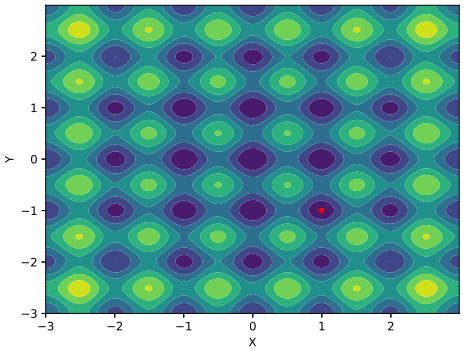
(e) Evolução do valor médio da função com mais repetições.



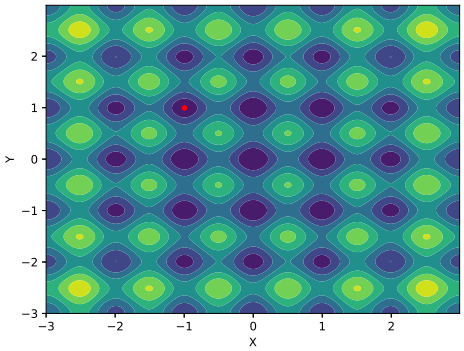
(f) Evolução do valor ótimo da função com mais repetições.

Figura 3: Otimização da função Schaffer N. 2.

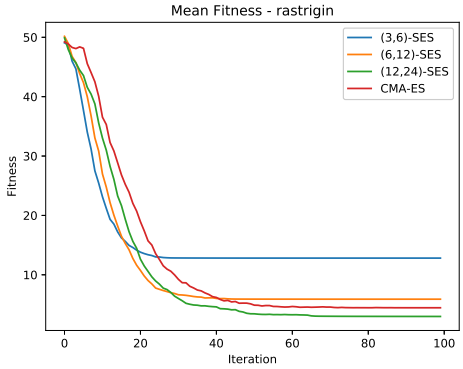
Com a função Schaffer N. 2, os algorítmos custaram mais para convergir a um valor ótimo, especialmente no caso do CMA-ES. Sendo assim, o teste foi realizado novamente, dessa vez com 500 repetições, cada uma com 200 iterações. Os resultados finais mostram os algorítmos (12,24)-SES e (6,12)-SES encontrando valores ótimos melhores que o CMA-ES, e o (3,6)-SES novamente convergindo pra um mínimo local significativamente pior que os demais testes com SES. Nesse caso, nenhum dos algorítmos chegou exatamente no mínimo global da função, que seria $f(0,0) = 0$, embora o (12,24)-SES e o (6,12)-SES chegaram a valores próximos.



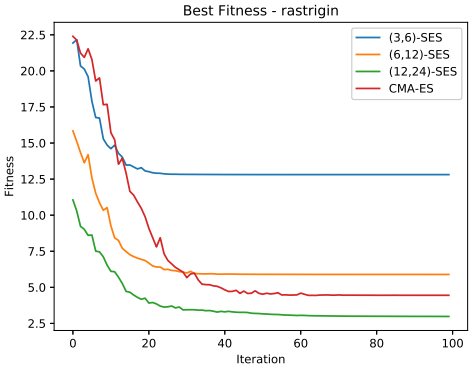
(a) Otimização com (12,24)-SES.



(b) Otimização com CMA-ES.



(c) Evolução do valor médio da função.



(d) Evolução do valor ótimo da função.

Figura 4: Otimização da função Rastrigin 2D.

O teste da função Rastrigin 2D novamente mostrou o (12,24)-SES obtendo o melhor valor ótimo, seguido pelo CMA-ES, (6,12)-SES e (3,6)-SES. Novamente, nenhum dos algorítmos encontrou o mínimo global da função, na qual $f(0,0) = 0$, mas nesse caso, todos encontraram valores relativamente distantes desse mínimo global, dado que o melhor mínimo encontrado teve valor superior a 2,5.

Diferente das primeiras duas funções, as duas últimas têm várias regiões de mínimos locais com valores não muito diferentes do mínimo global, o que pode ser observado pelas curvas de nível das funções ilustradas nas figuras. Isso explica a dificuldade dos algorítmos de acharem o mínimo global. No caso do Rastrigin 2D, essa dificuldade é mais evidente, o que pode ser explicado pelo número elevado de mínimos locais existentes.

Em geral, observando a evolução de alguns testes do algorítmo (12,24)-SES e do CMA-ES, foi observado que o SES tende a convergir seus pontos mais rapidamente, enquanto o CMA-ES mantém seus pontos dispersos por mais tempo. Isso pode ajudá-lo a não ficar preso tão facilmente em mínimos locais, pois o CMA-ES acaba explorando mais dessa forma. A desvantagem, nesse caso, é um tempo maior para convergência do algorítmo. Finalmente, é importante lembrar que o algorítmo CMA-ES foi executado com $\mu = 3$ e $\lambda = 6$. Considerando essa desvantagem significativa com relação aos testes usando (6,12)-SES e (12,24)-SES, observa-se que o CMA-ES teve um desempenho notável, encontrando mínimos significativamente menores que o (3,6)-SES em quase todos os testes realizados.