**Instituto Tecnológico de Aeronáutica - ITA**
**CT-213 - Inteligência Artificial aplicada à Robótica Móvel**
**Aluno**: Ulisses Lopes da Silva

**Relatório do Laboratório 5 - Estratégias Evolutivas**

---

## 1 Breve Explicação em Alto Nível da Implementação

### 1.1 Estratégia Evolutiva Simples

A implementação da Estratégia Evolutiva Simples (SES) desenvolvida neste laboratório baseia-se em um processo iterativo de otimização, no qual uma população de soluções candidatas é gerada, avaliada e usada para atualizar os parâmetros de uma distribuição gaussiana multivariada. A ideia central é evoluir ao longo das gerações tanto a média quanto a matriz de covariância dessa distribuição, que modela onde se espera encontrar boas soluções no espaço de busca.

Inicialmente, a população é amostrada a partir de uma distribuição normal multivariada com média $\mathbf{m}^{(0)}$ e matriz de covariância $\mathbf{C}^{(0)}$. Cada indivíduo (amostra) gerado é avaliado pela função de custo, e os $\mu$ melhores indivíduos são selecionados com base em seus desempenhos (menores valores de fitness, no caso de minimização). Estes indivíduos são chamados de pais e são utilizados para atualizar a distribuição para a próxima geração.

A nova média $\mathbf{m}^{(g+1)}$ é obtida pela média aritmética dos $\mu$ melhores indivíduos, e a nova matriz de covariância $\mathbf{C}^{(g+1)}$ é calculada com base na dispersão dos pais em torno da nova média, conforme a equação:

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

Essas atualizações permitem que a distribuição evolua de forma adaptativa, promovendo um equilíbrio entre exploração e convergência.

O algoritmo foi implementado com base na classe `SimpleEvolutionStrategy`, composta principalmente pelos métodos `ask()`, que retorna a população da geração atual, e `tell()`, responsável pela atualização da média e da matriz de covariância. A implementação aproveita os recursos vetorizados do NumPy para eficiência computacional. Embora mais simples que métodos avançados como o CMA-ES, a SES é eficaz para funções com topologia moderada e serve como base para comparações em estudos de benchmark.

## 2 Figuras Comprovando Funcionamento do Código

### 2.1 Função *Translated Sphere*

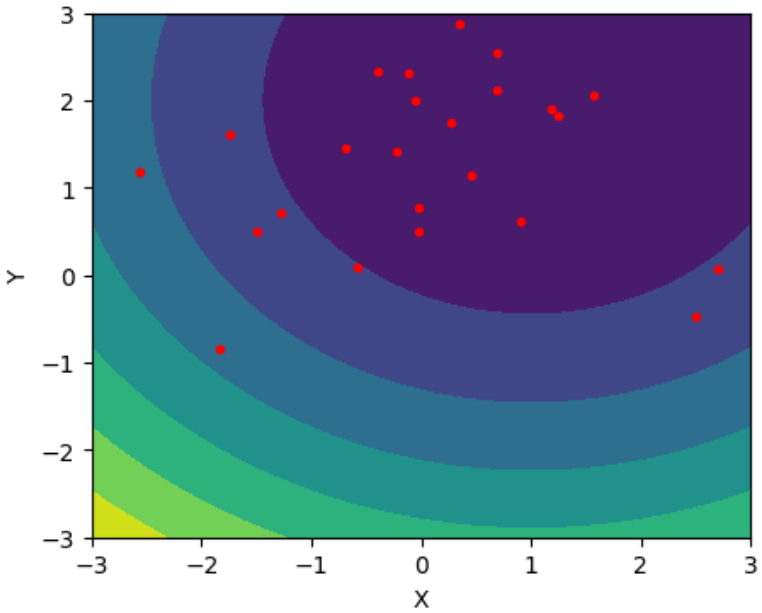


Fig. 1: Partículas amostrais iniciais geradas aleatoriamente para o teste da SES, com a função Translated Sphere

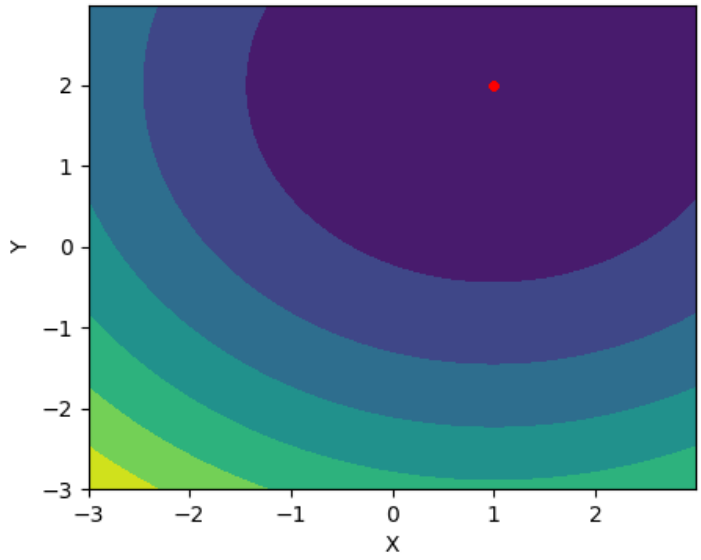


Fig. 2: Resultado obtido pela SES, com a função Translated Sphere

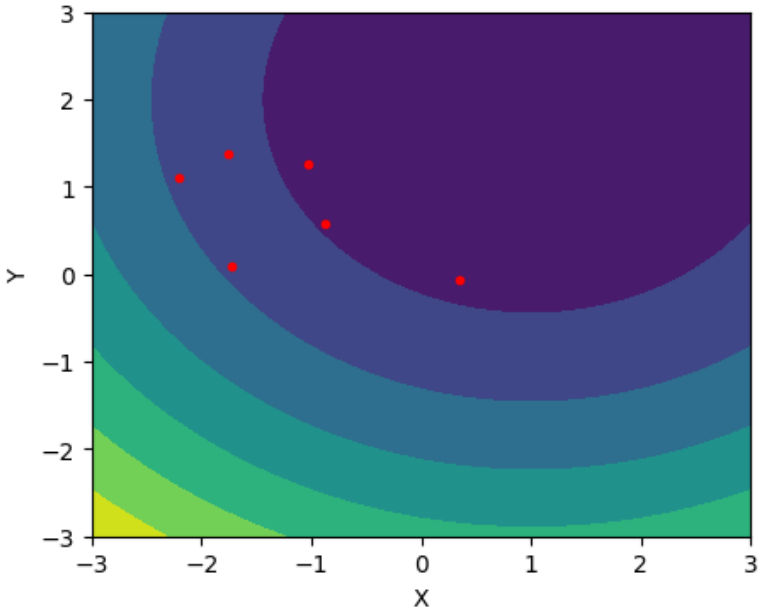


Fig. 3: Partículas amostrais iniciais geradas aleatoriamente para o teste da CMAES, com a função Translated Sphere

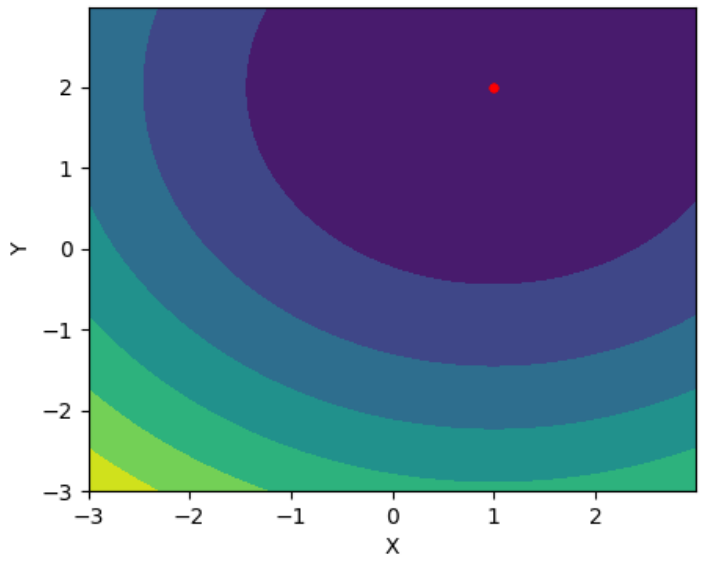


Fig. 4: Resultado obtido pela CMAES, com a função Translated Sphere

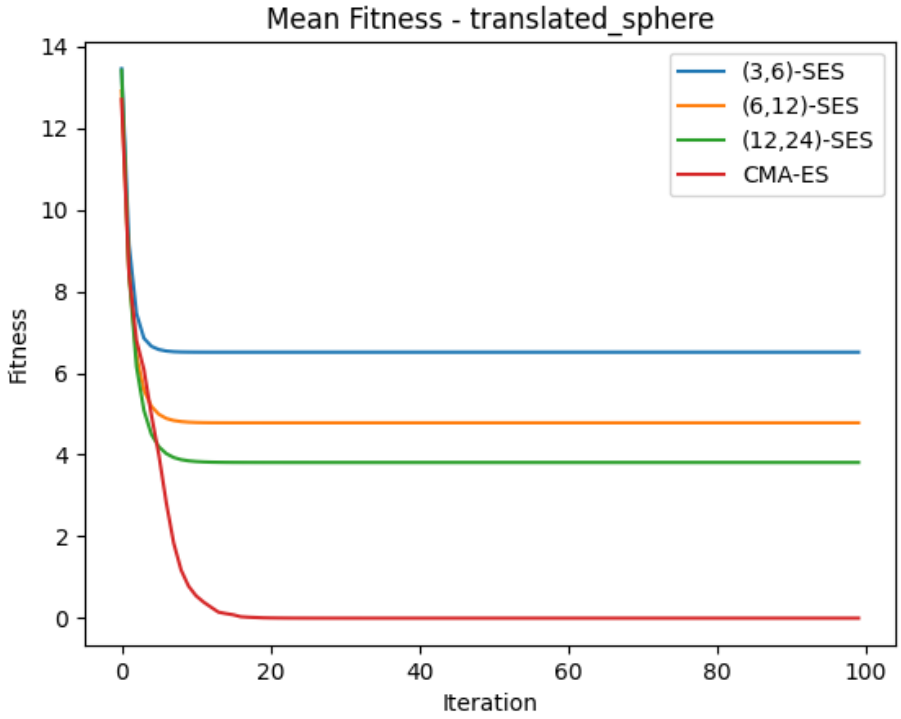


Fig. 5: *Fitness* médio das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

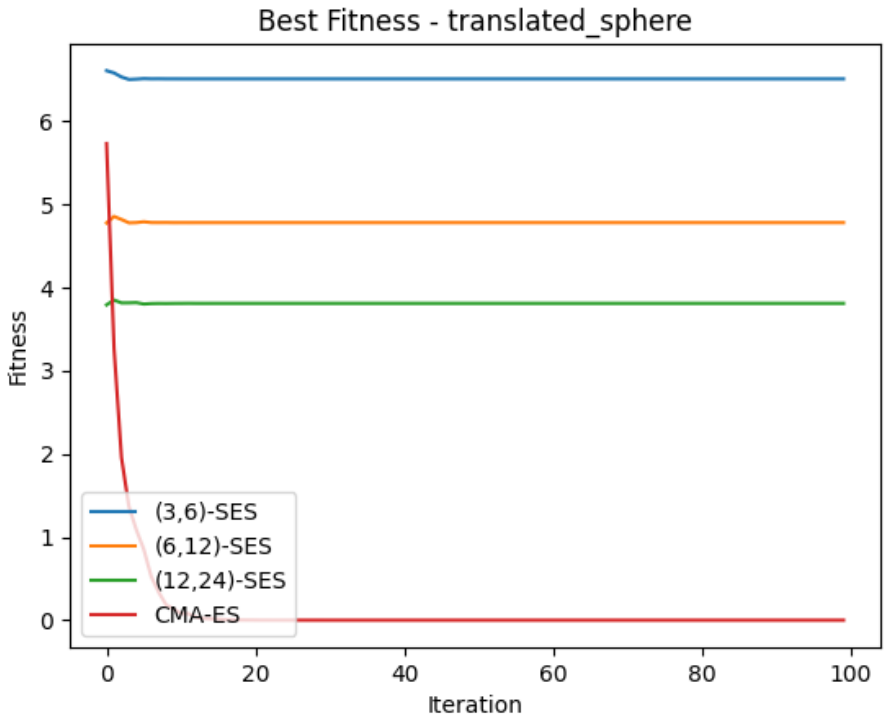


Fig. 6: Melhor *fitness* das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

## 2.2 Função Ackley

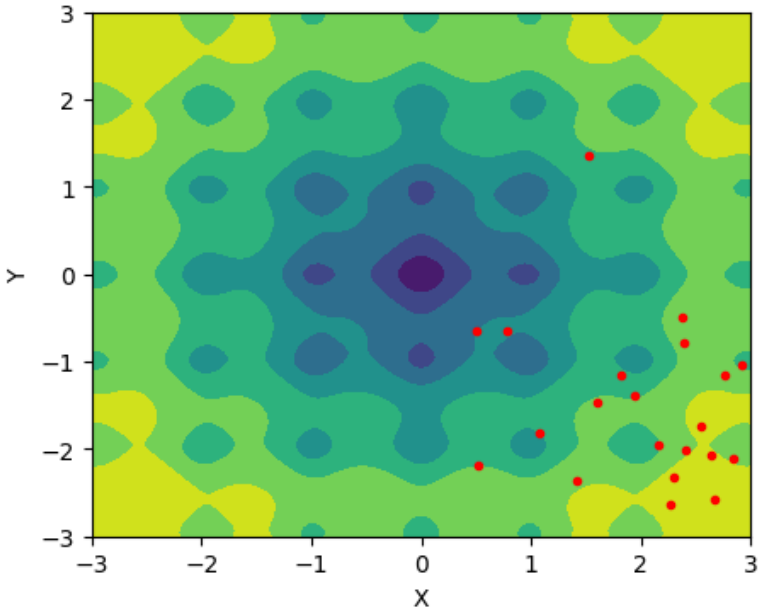


Fig. 7: Partículas amostrais iniciais geradas aleatoriamente para o teste da SES, com a função Ackley

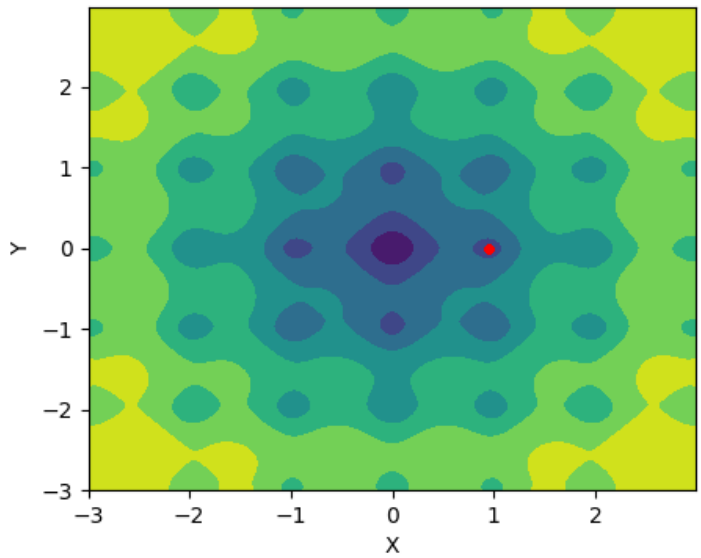


Fig. 8: Resultado obtido pela SES, com a função Ackley

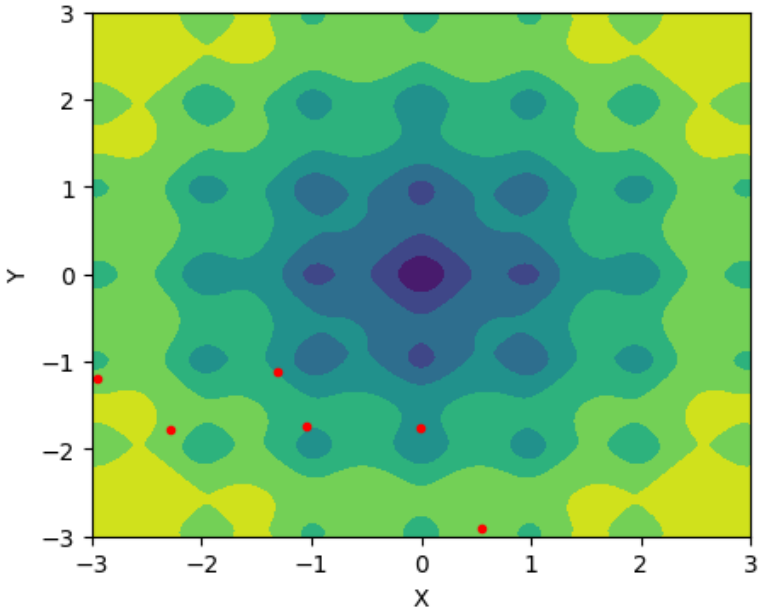


Fig. 9: Partículas amostrais iniciais geradas aleatoriamente para o teste da CMAES, com a função Ackley

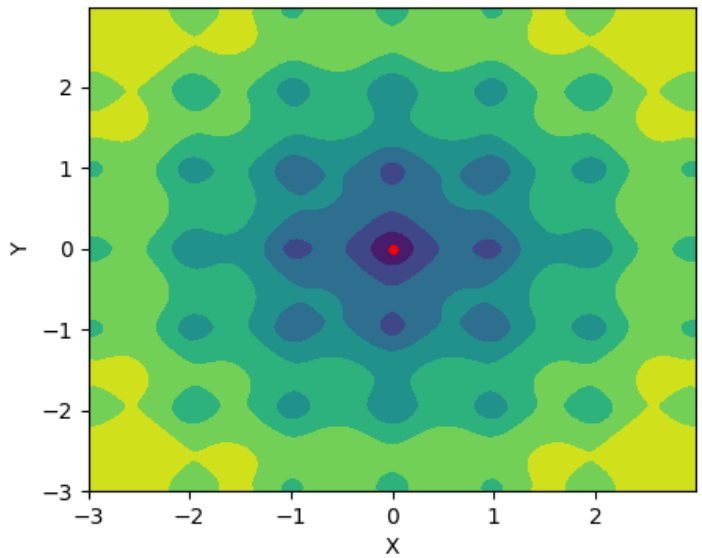


Fig. 10: Resultado obtido pela CMAES, com a função Ackley

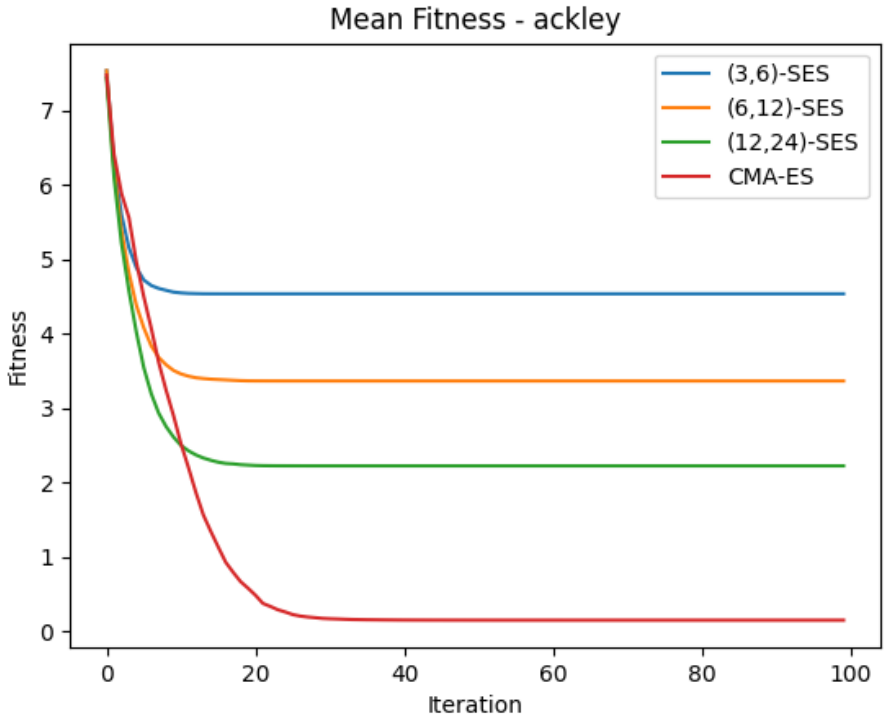


Fig. 11: *Fitness* médio das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

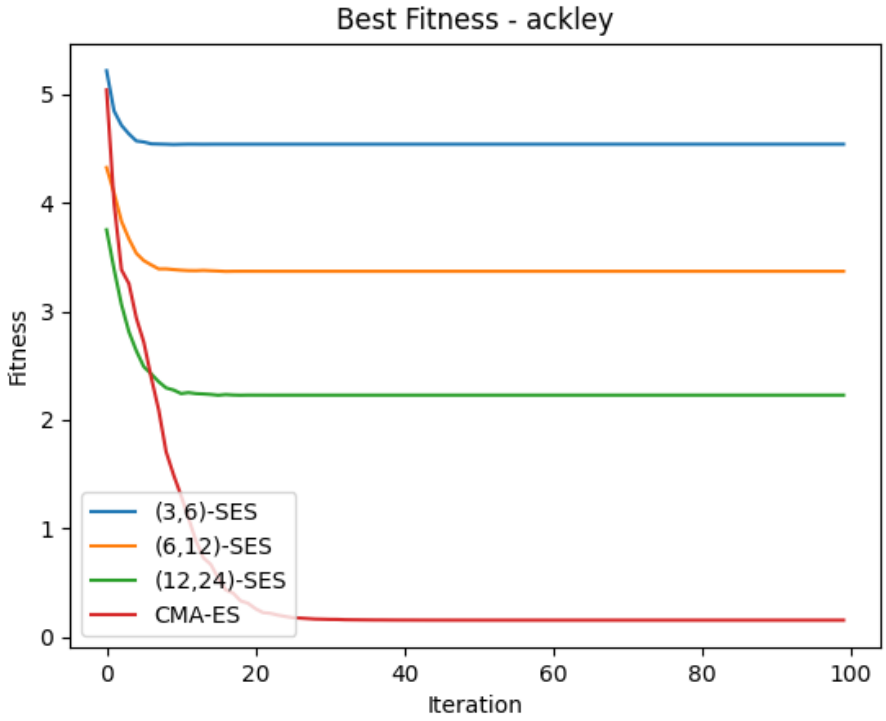


Fig. 12: Melhor *fitness* das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

### 2.3 Função Rastrigin

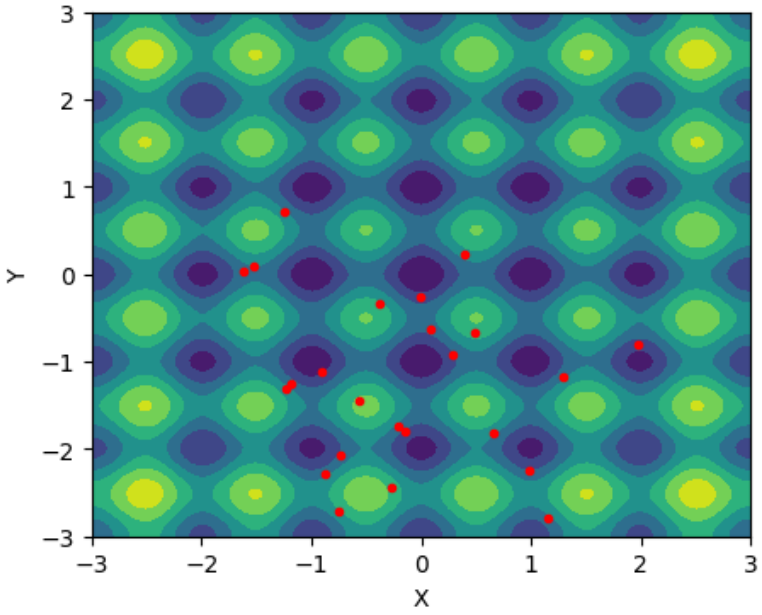


Fig. 13: Partículas amostrais iniciais geradas aleatoriamente para o teste da SES, com a função Rastrigin

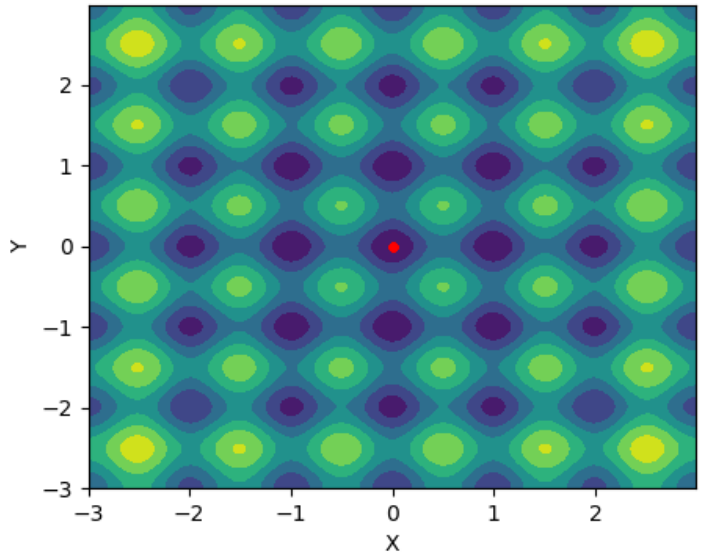


Fig. 14: Resultado obtido pela SES, com a função Rastrigin

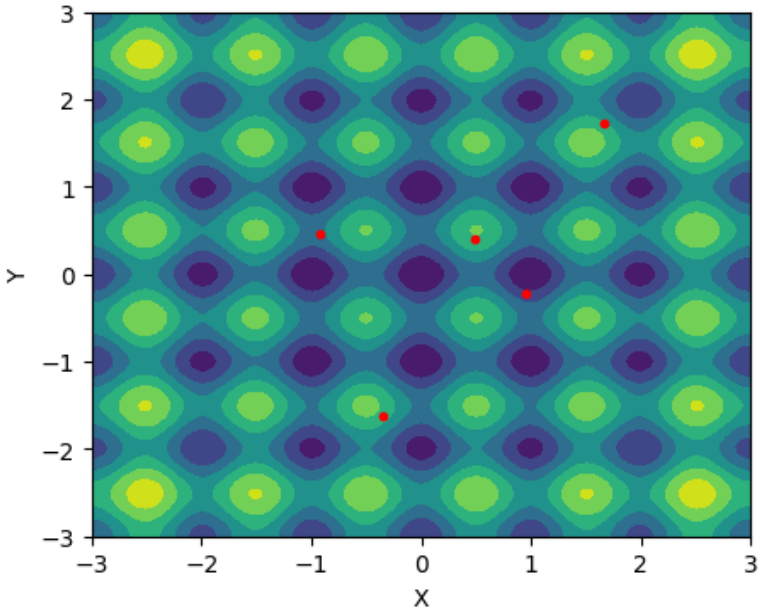


Fig. 15: Partículas amostrais iniciais geradas aleatoriamente para o teste da CMAES, com a função Rastrigin

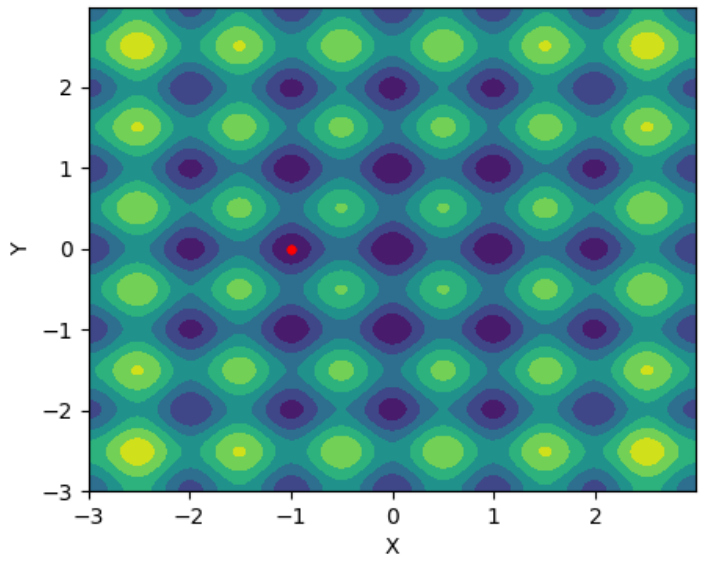


Fig. 16: Resultado obtido pela CMAES, com a função Rastrigin

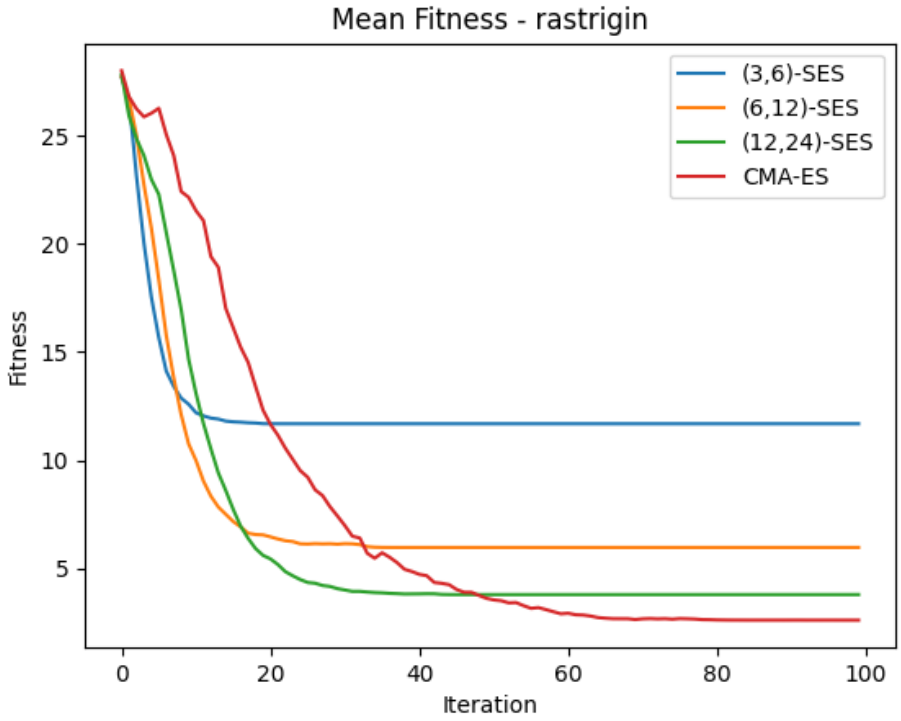


Fig. 17: *Fitness* médio das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

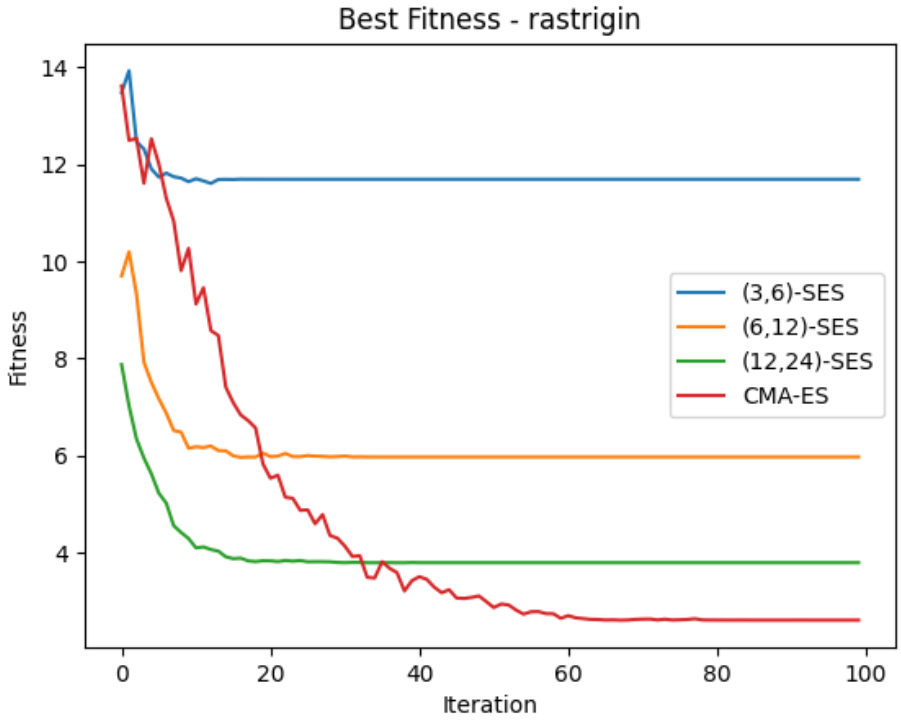


Fig. 18: Melhor *fitness* das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

## 2.4 Função Schaffer

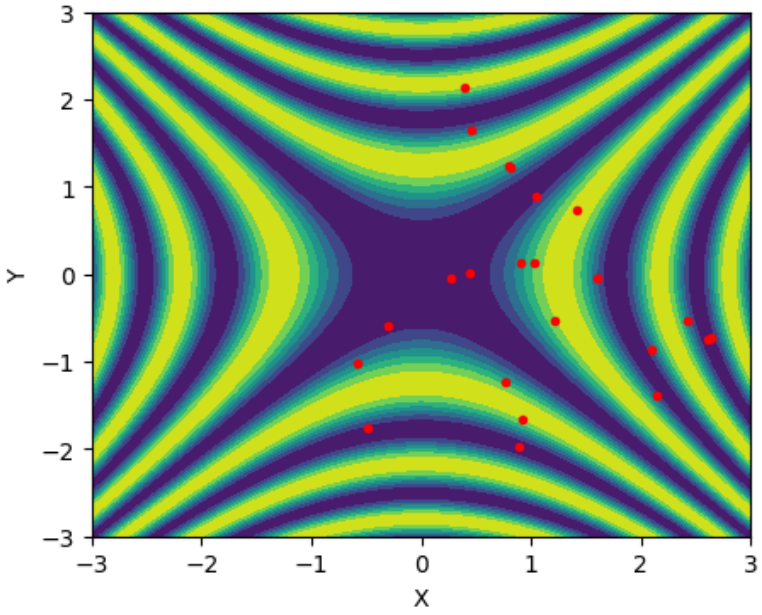


Fig. 19: Partículas amostrais iniciais geradas aleatoriamente para o teste da SES, com a função Schaffer

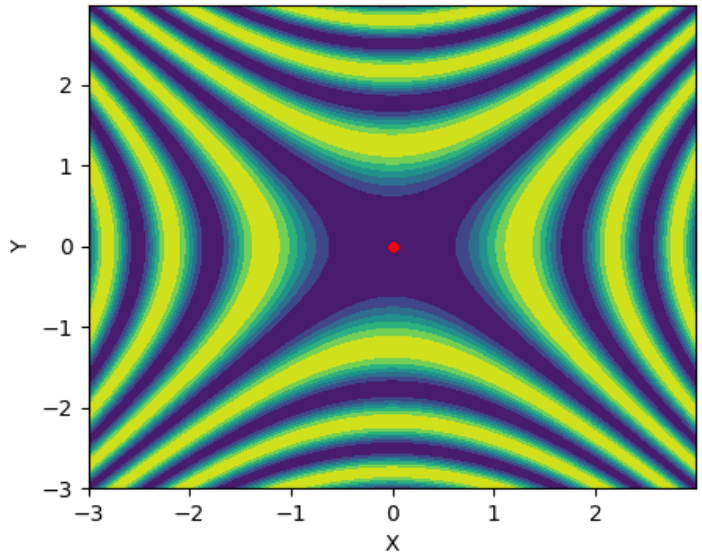


Fig. 20: Resultado obtido pela SES, com a função Schaffer

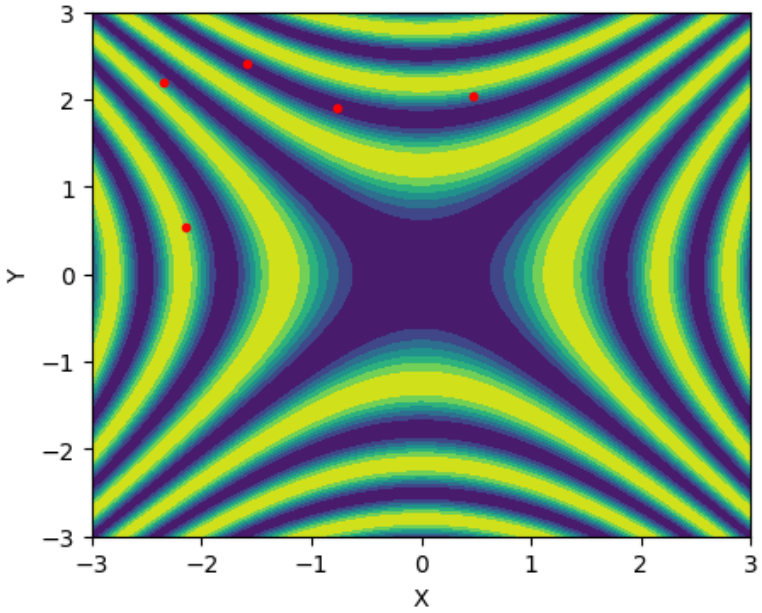


Fig. 21: Partículas amostrais iniciais geradas aleatoriamente para o teste da CMAES, com a função Schaffer

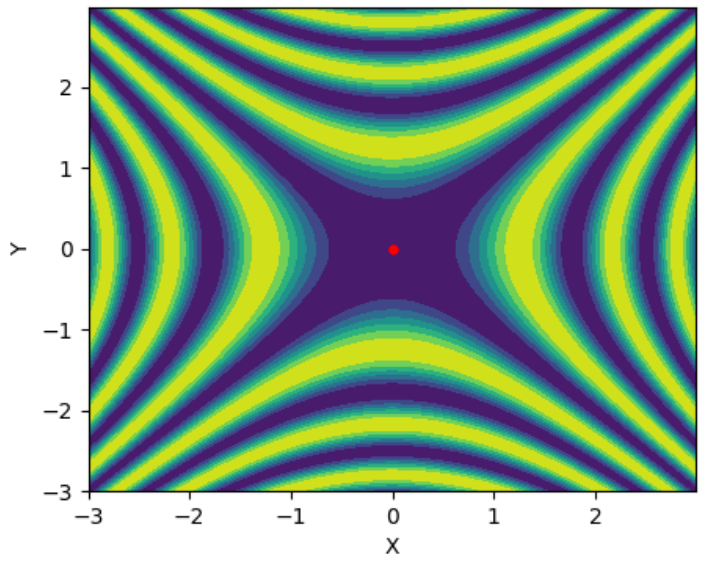


Fig. 22: Resultado obtido pela CMAES, com a função Schaffer

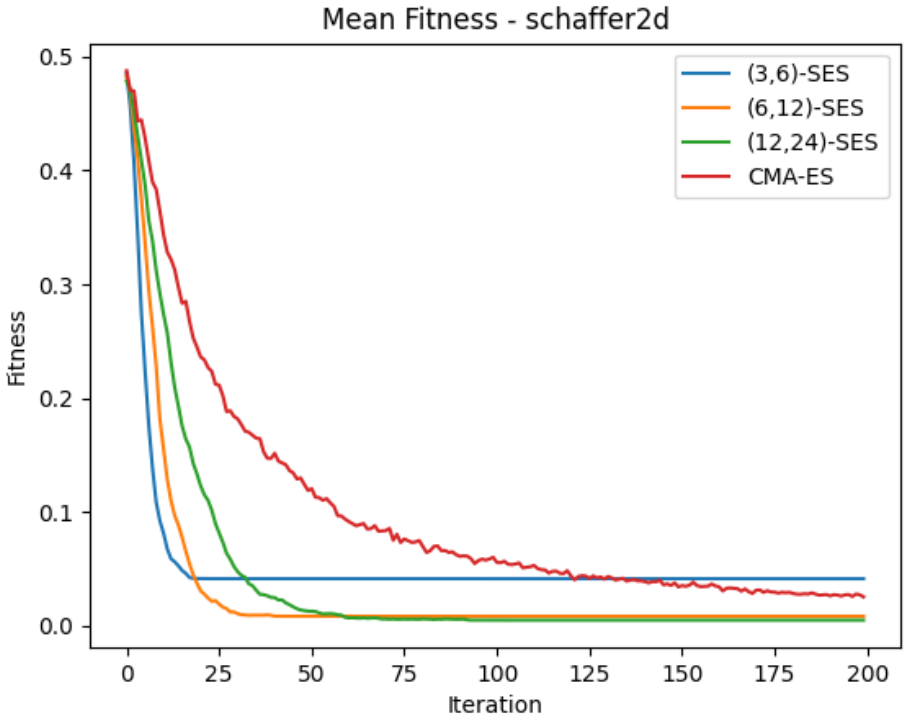


Fig. 23: *Fitness* médio das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

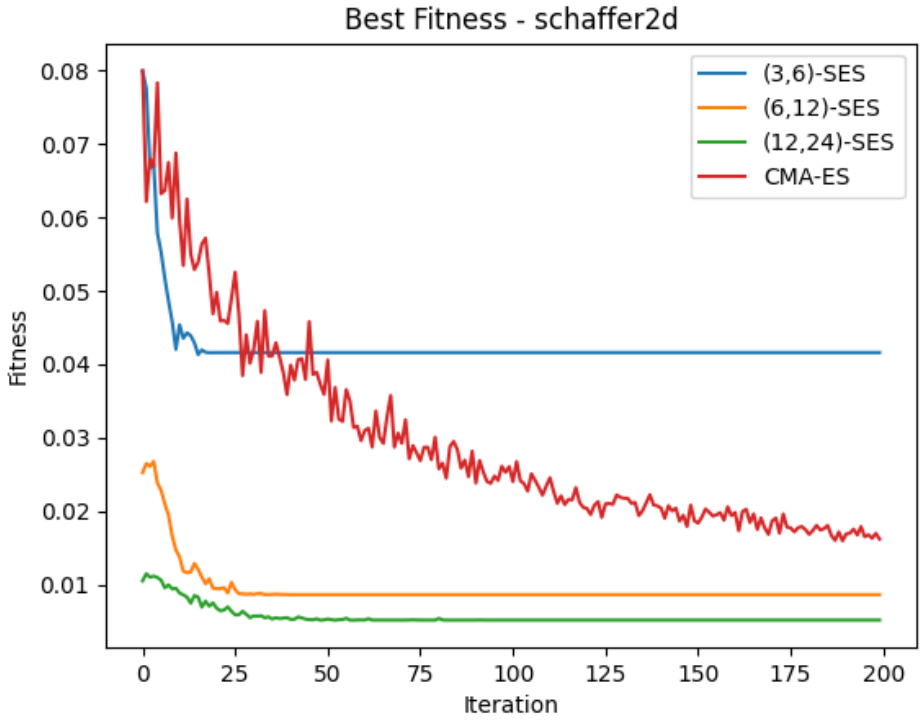


Fig. 24: Melhor *fitness* das amostras em cada geração (calcula-se uma média entre todas as simulações de Monte Carlo)

## 3 Discussões e Conclusões

- **Por que você acha que os resultados são diferentes para cada função?**

  Os resultados variaram bastante de uma função para outra, e isso tem relação direta com a complexidade do formato de cada função objetivo. A *Translated Sphere*, por exemplo, apresentou uma superfície convexa e unimodal, o que favoreceu a convergência rápida da Estratégia Evolutiva Simples (SES), mesmo com sua estrutura mais básica. Em contrapartida, funções como a *Rastrigin* e a *Schaffer* mostraram ser muito mais desafiadoras, por possuírem diversos mínimos locais. Nesses casos, algoritmos mais simples, que não têm mecanismos sofisticados para escapar de mínimos locais, acabam enfrentando dificuldades para alcançar o ótimo global. Isso mostra que tanto a estrutura da função quanto a complexidade do algoritmo afetam diretamente a eficiência e a trajetória da convergência.

  Outro ponto importante que foi observado foi a variação dos resultados entre diferentes execuções. Em algumas rodagens, os algoritmos convergiram para bons valores; em outras, acabaram presos em mínimos locais. Esse comportamento instável parece estar ligado, principalmente, à forma como as amostras iniciais são geradas. A SES, por exemplo, começava com o dobro de partículas em comparação com o CMA-ES. Por conta disso, havia uma maior chance de que, por pura aleatoriedade, algumas dessas partículas fossem geradas mais próximas do ótimo global, o que acabava facilitando a convergência. Quando isso

acontecia, os resultados eram bastante satisfatórios. Por outro lado, se muitas partículas eram geradas longe do ótimo, o algoritmo frequentemente convergia para regiões pouco desejáveis. A quantidade de partículas também fazia diferença: o SES, tendo mais partículas por geração, em geral conseguia explorar mais regiões e encontrar boas soluções — desde que essas regiões estivessem razoavelmente próximas. Em execuções onde tanto SES quanto CMA-ES começavam longe da solução ideal, ambos os algoritmos enfrentavam dificuldades. Em muitos casos, as partículas nem sequer permaneciam dentro da área visível do gráfico. A sensação era a de que os mínimos locais funcionavam como "buracos negros", atraindo a maioria das partículas e distorcendo a convergência.

- **Comente de forma sucinta sobre os resultados para cada um dos algoritmos e das funções, principalmente sobre questões como convergência, incluindo sobre convergência para mínimo local.**

  O SES, por ser um algoritmo mais simples, teve bom desempenho nas funções mais suaves e com formato simples, como a *Translated Sphere*. A convergência foi rápida e estável, desde que o ponto de partida não estivesse muito longe do ótimo global. Nas funções mais complexas, como *Ackley*, *Schaffer* e *Rastrigin*, o SES teve mais dificuldade, muitas vezes ficando preso em mínimos locais. Isso se deve à ausência de estratégias internas de adaptação ou mecanismos mais refinados de seleção e recombinação, o que torna o algoritmo mais suscetível às armadilhas do espaço de busca.

  Já o CMA-ES mostrou resultados bem mais robustos em funções complicadas. Sua capacidade de adaptar a matriz de covariância com mais inteligência e de utilizar estratégias de recombinação eficazes ajudou bastante na exploração do espaço de busca, o que, na prática, levou a melhores convergências — geralmente mais próximas dos ótimos globais. Ainda assim, percebi que o CMA-ES também é sensível à escolha inicial: quando as partículas eram muito dispersas ou concentradas em regiões irrelevantes, a convergência era menos satisfatória.

Com base nessas observações, é possível concluir que o CMA-ES é mais adequado para problemas de otimização com funções complexas, especialmente aquelas que apresentam múltiplos mínimos locais. No entanto, seu desempenho pode ser ainda melhor se for adotada uma estratégia mais inteligente na escolha das posições iniciais, com locais "mais promissores". O SES, por sua vez, apesar de mais simples e computacionalmente mais leve, funciona bem em funções menos complicadas, com menos picos e vales. Mesmo assim, os gráficos e os resultados das simulações indicam que, com o passar das gerações, o CMA-ES tende a superar o SES, alcançando valores de *fitness* consistentemente melhores.