Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT-213
Aluno:

Relatório do Laboratório 5 - Estratégias Evolutivas

1. Breve Explicação em Alto Nível da Implementação

1.1. Estratégia Evolutiva Simples

Primeiramente, inicializa-se as variáveis m, a média dos melhores valores, C, a matriz de covariância, mu, o número de elementos do vetor de melhores amostras, population_size, o tamanho da população e o vetor samples, que carrega a população que será otimizada, implementado pela função np.random.multivariate_normal do numpy e de parâmetros média, matriz de covariância e tamanho da população. Assim, dentro da classe Simple Evolution Strategy, depois de inicializar as variáveis na _init_, temos uma função da classe chamada ask, que apenas retorna as samples. Ainda temos a função tell, responsável por atualizar as variáveis. Primeiramente, se seleciona os índices das mu melhores samples ordenadas, comparando o fitnesses. Assim, pode-se determinar os pais da amostra, que são as mu melhores, que serão utilizados para calcular a média dos melhores. Após isso, atualiza-se a matriz de covariância, o que pode ser feito de diferentes modos, mas que no laboratório seguiu a fórmula específica:

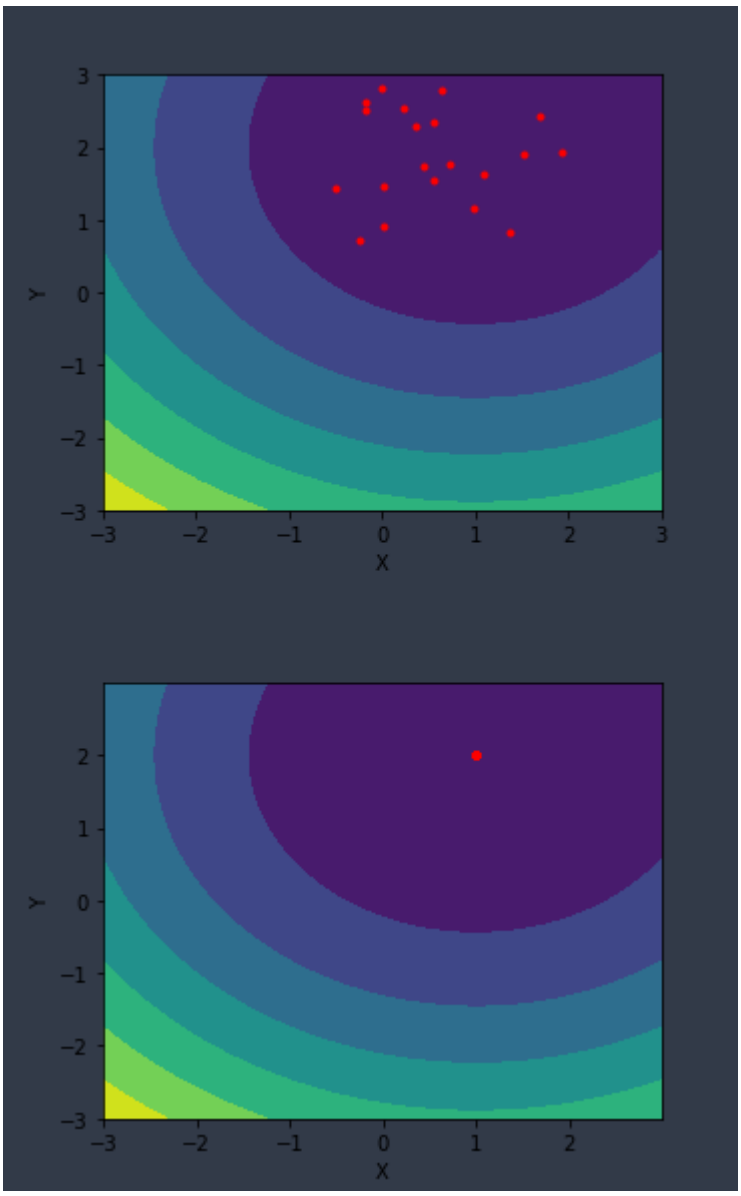
$$C^{(g+1)} = \frac{1}{\mu} \sum_{i=1}^{\mu} \left(s_{i:\lambda}^{(g+1)} - m^{(g)} \right) \left(s_{i:\lambda}^{(g+1)} - m^{(g)} \right)^T$$

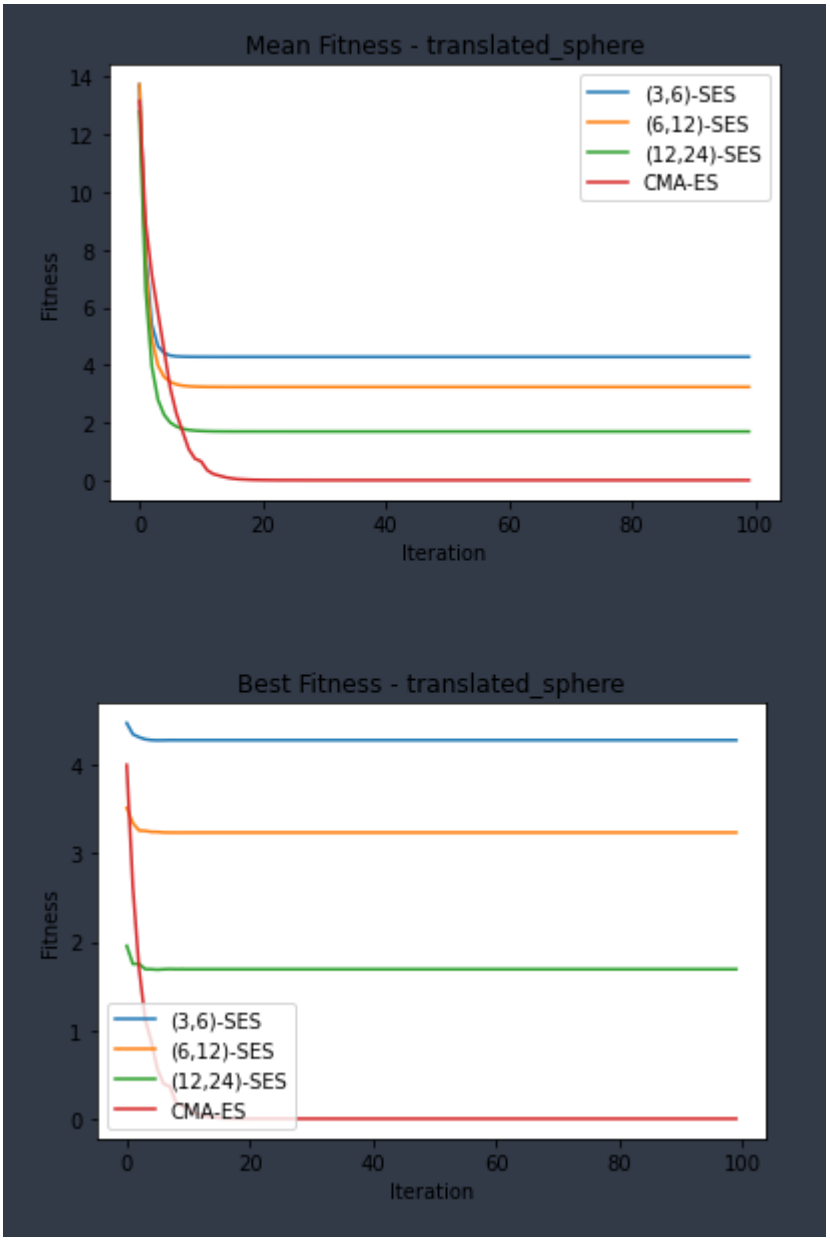
Desse modo, atualiza-se o vetor sample com os novos parâmetros m,C e population_size.

Conhecendo essas funções da classe Simple Evolution Strategy, podemos implementar essa estratégia. Enquanto não assar o número de interação, chama-se o self.sample pela função já explicada ask e define-se o fitnesses segundo um certa function. Após isso, chama-se a função tell para atualizar os valores dos parâmetros segundo a nova função fitnesses, até acabar o número máximo de iterações. No fim, escolhe-se a melhor sample.

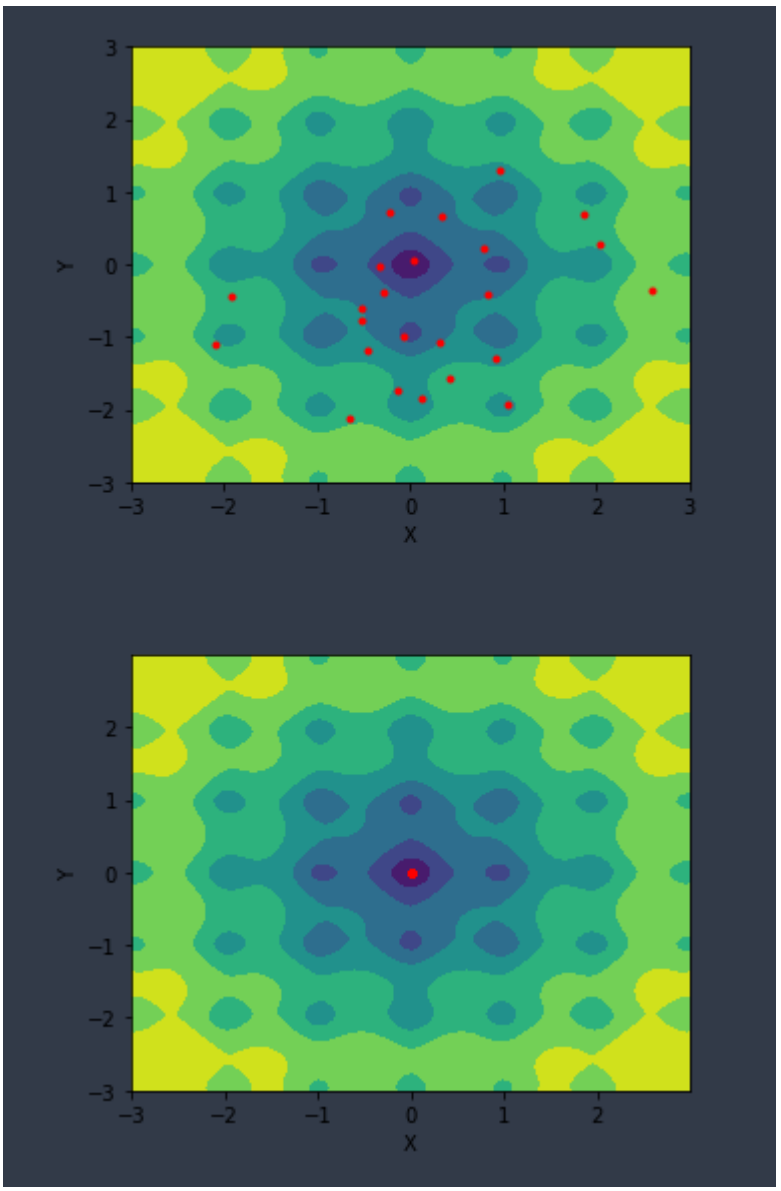
2. Figuras Comprovando Funcionamento do Código

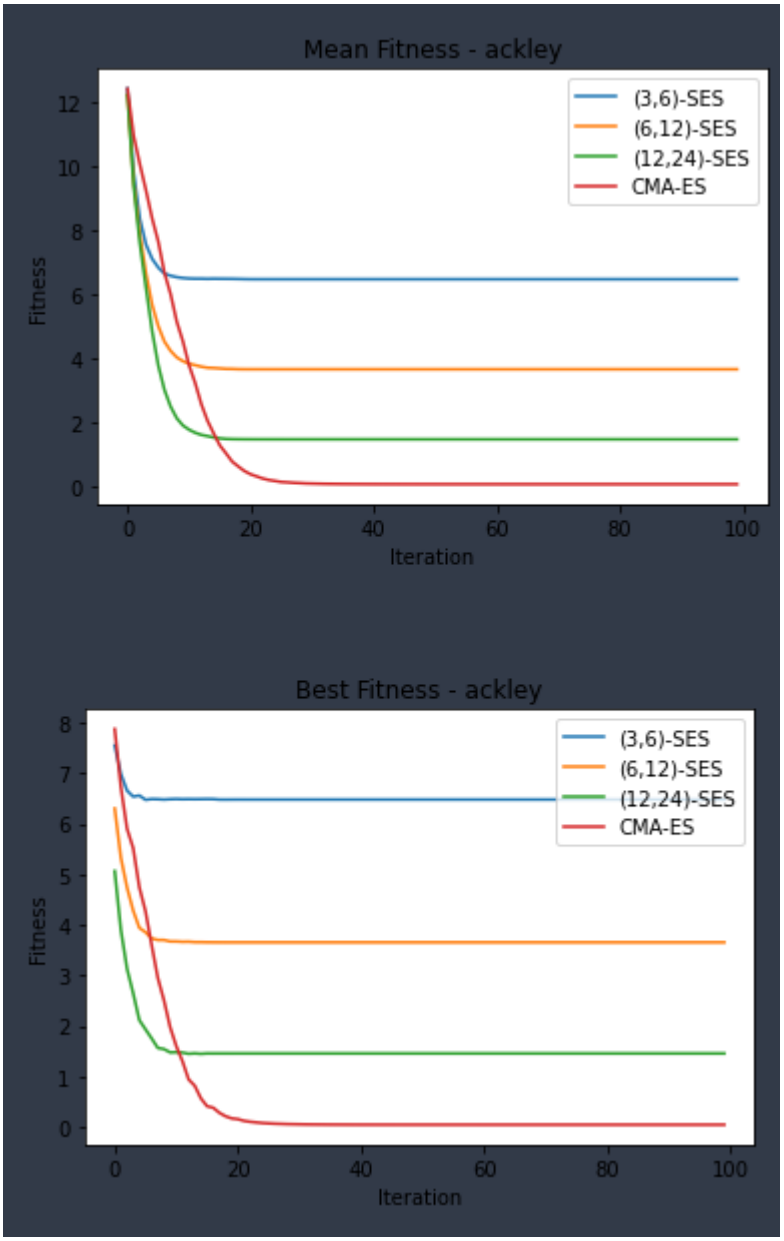
2.1. Função *Translated Sphere*


3
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
3
X
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
X

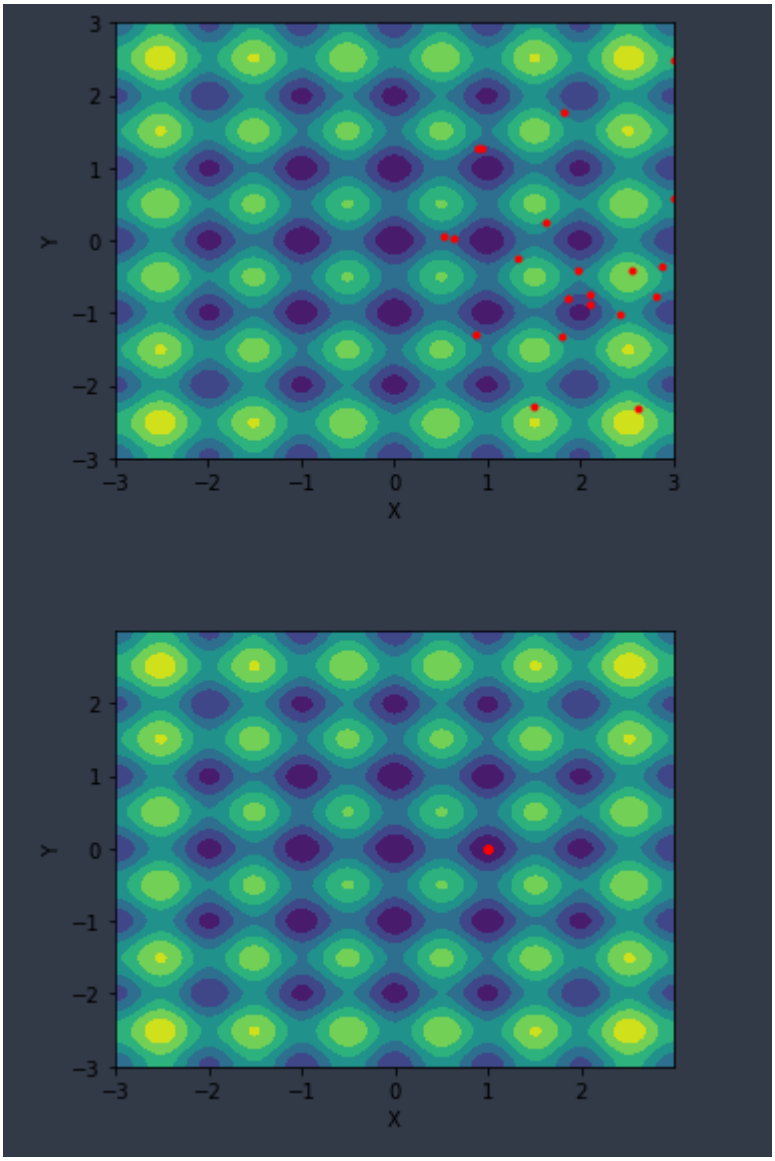


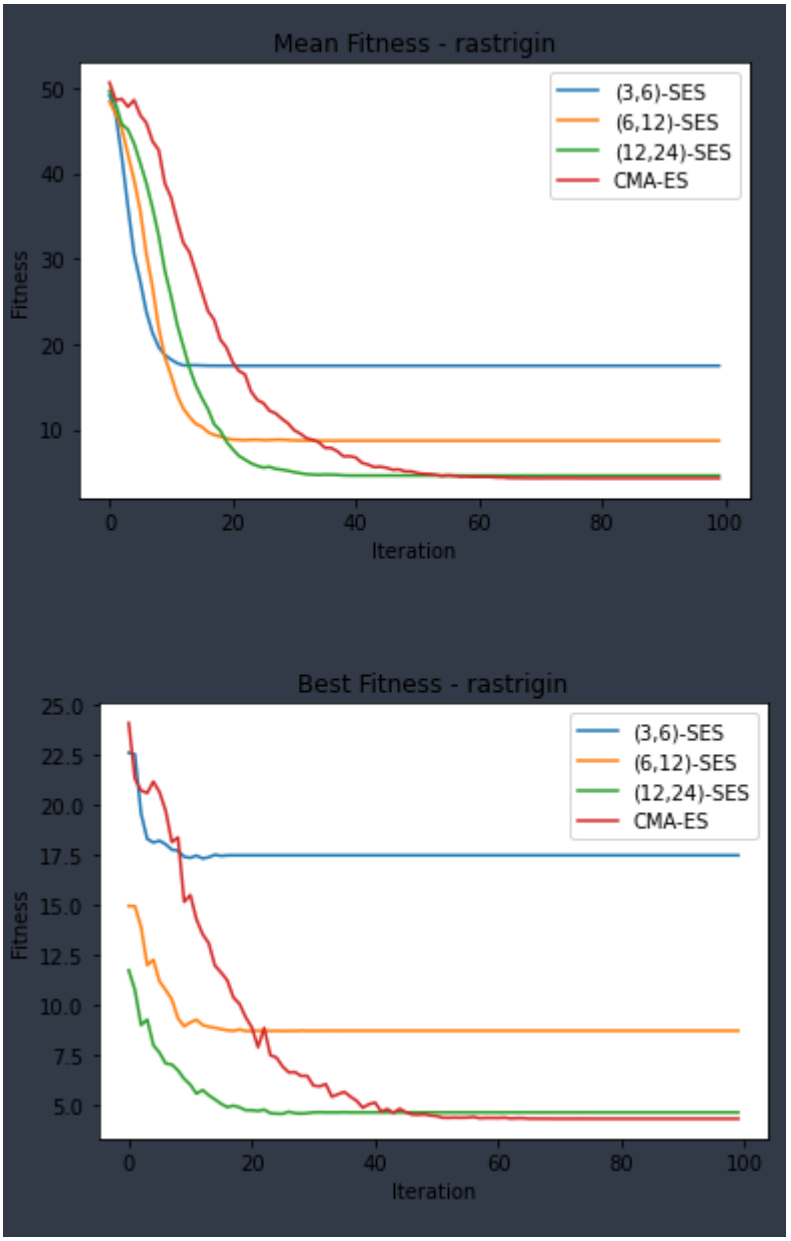
2.2. Função Ackley


3
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
3
X
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
X

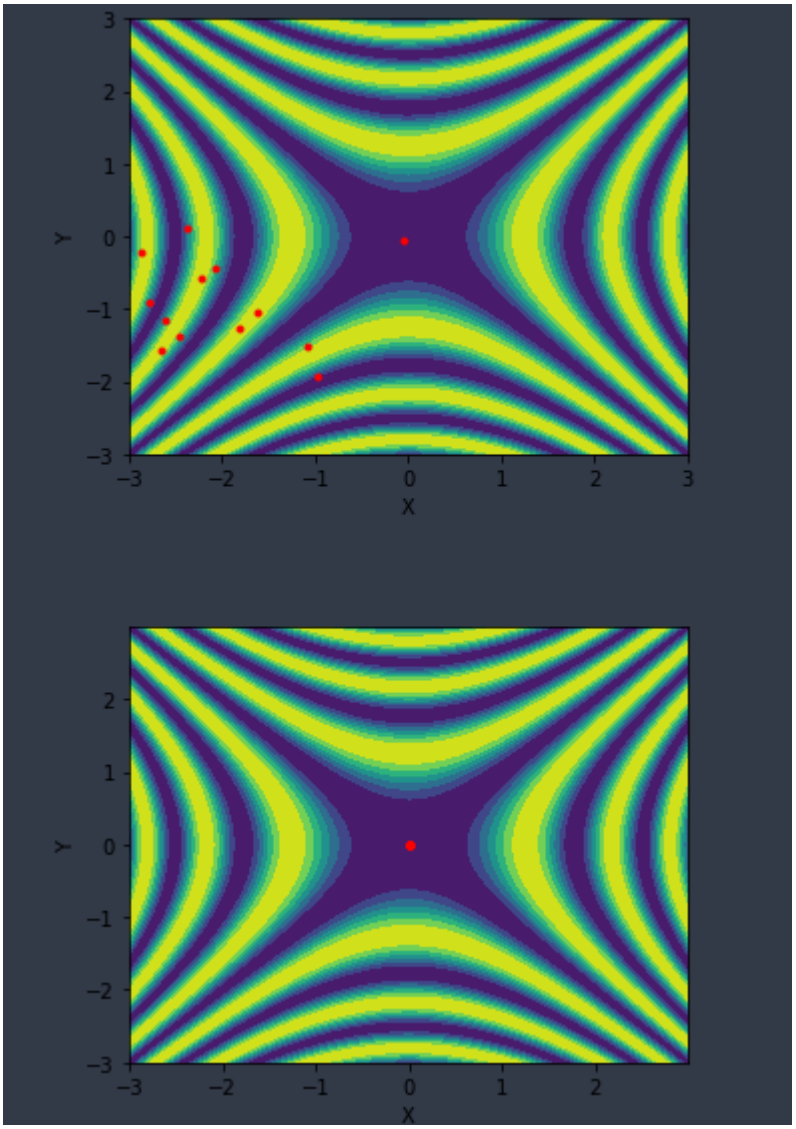


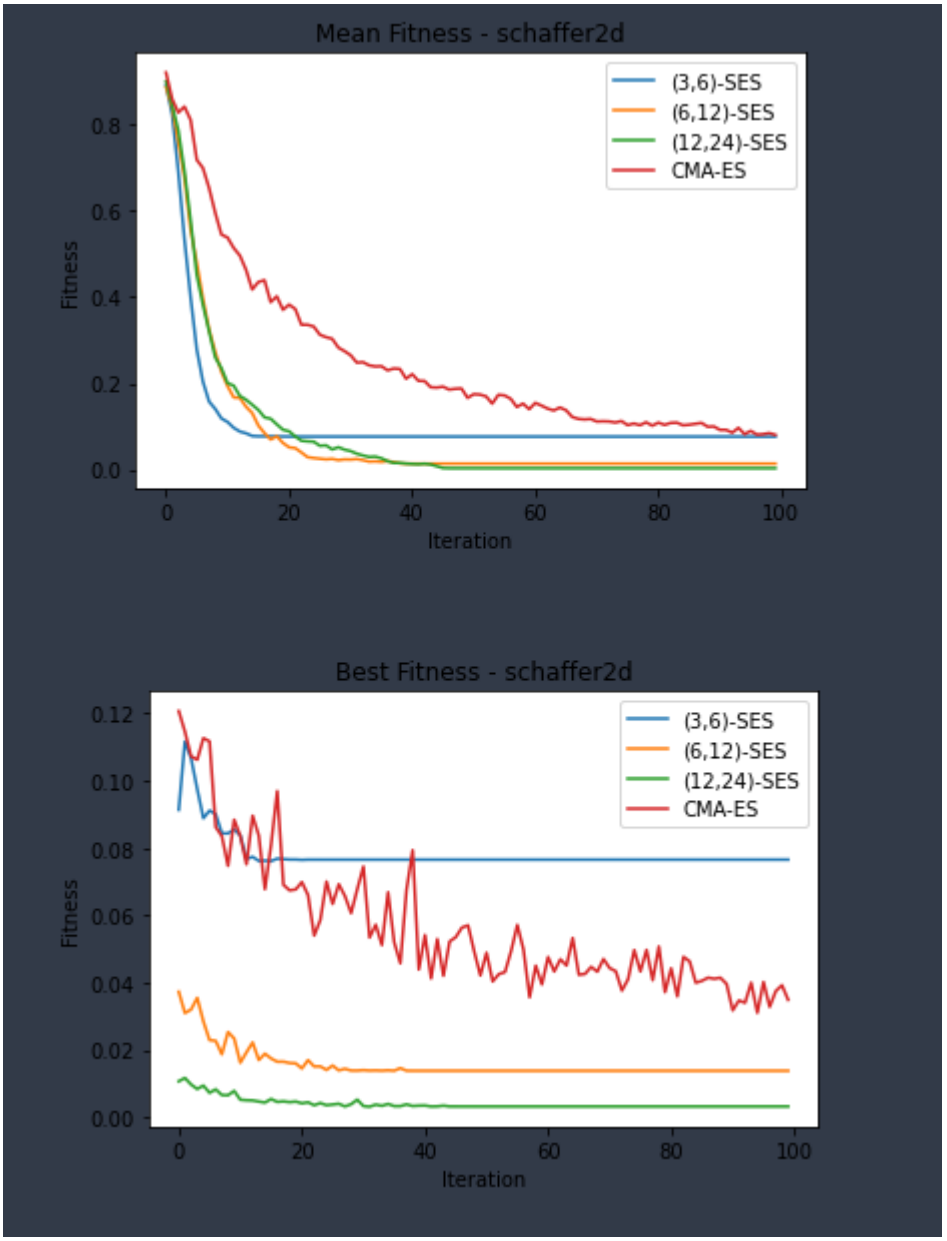
2.3. Função Rastrigin


3
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
3
X
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
X



2.4. Função Schaffer


3
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
3
X
2
1
0
−1
−2
−3
Y
−3
−2
−1
0
1
2
X



3. Discussões e Conclusões

- Por que você acha que os resultados são diferentes para cada função?
  O CMS-ES é um algoritmo muito mais complexo e otimizado que o SES implementado, que é uma implementação bem simples de estratégia evolutiva, com uma atualização de matriz de covariância e com uma média dos melhores bem simples. Evidentemente, dentre os SES, a com maior número de parâmetros tende a ter um melhor fitness, mas fica mais lenta. De modo especial, em alguns casos muito específicos o CMA-ES pode ser pior que algoritmos de otimização mais simples, pois a estrutura desses algoritmos simples pode casar muito com alguns problema específicos.
- 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.

Todos os algoritmos conseguem em todas as funções convergir para um mínimo global um mínimo global, bastava que se repetisse algumas vezes o programa, pois o resultado mudava de acordo com a sample inicial, podendo ficar enviezada. De modo especial, a Função Ackley e a Função Rastrigin tinham uma alta tendência de ficarem presas num mínimo local.