

# Relatório do Laboratório 5:

## Estratégias Evolutivas

Isabelle Ferreira de Oliveira  
CT-213 - Engenharia da Computação 2020  
Instituto Tecnológico de Aeronáutica (ITA)  
São José dos Campos, Brasil  
isabelle.ferreira3000@gmail.com

**Resumo**—Esse relatório documenta a implementação de uma estratégia evolutiva simples e comparação de seu desempenho com o de CMA-ES em funções usadas como *benchmark* para algoritmos de otimização. Essas funções utilizadas eram uma esfera transladada, e as funções de Ackley, Schaffer 2 e Rastrigin 2D.

**Index Terms**—Estratégias evolutivas, CMA-ES, *benchmark*, Ackley, Schaffer 2, Rastrigin 2D

### I. INTRODUÇÃO

Otimização consiste em encontrar o mínimo (ou máximo) de uma função, ou seja, encontrar o conjunto de parâmetros que levem essa função ao seu mínimo (ou máximo). Também pode ser visto como encontrar a melhor solução dentre todas as soluções viáveis. Nesses problemas de otimização também é possível haver restrições acerca dos parâmetros que serão analisados.

Dentre os mais diversos tipos de algoritmos de otimização, existem os métodos baseado em estratégias evolutivas, métodos esses inspirados na evolução natural das espécies. Esses métodos seguem a ideia de: dada uma população de possíveis soluções, aplica-se uma função para medir a qualidade dessas soluções candidatas. Essa qualidade quantitativa é chamada de *fitness*. Com base no *fitness*, é possível escolher as melhores soluções e, a partir delas, gerar mutações para se criar uma nova população. Esse processo é repetido até que a convergência chegue a um resultado satisfatório de otimização.

O pseudo-código geral de algoritmos utilizando estratégias evolutivas pode ser visto a seguir. Em seguida, será apresentado como esse algoritmo foi implementado no contexto do laboratório.

---

```
def evolution_strategy(J, m0, C0,
    hyperparams):
    mu, lambda = unwrap_hyperparams(hyperparams)
    m, C = m0, C0
    while not check_stopping_condition():
        population = multivariate_normal(m, C,
            lambda)
        population = sort_ascending(population, J)
        parents = population[0:mu]
        m = mean(parents)
    return population[0]
```

---

No pseudocódigo acima,  $J$  é a função para medir a qualidade das soluções candidatas;  $m0$  e  $C0$  são a média e a

matriz de covariância iniciais da população que será gerada, respectivamente;  $m$  e  $C$  são, de forma análoga, a média e a matriz de covariância atuais da população que será gerada, respectivamente; e  $\mu$  é o tamanho da população considerada como "melhores soluções até então".

### II. IMPLEMENTAÇÃO DO ALGORITMO

Na parte relativa a implementação do algoritmo PSO, era necessário preencher os construtores das classes *Particle* e *ParticleSwarmOptimization*, além dos códigos das funções *get\_best\_position()*, *get\_best\_value()*, *get\_position\_to\_evaluate()*, *advance\_generation()* e *notify\_evaluation()* da classe *ParticleSwarmOptimization*. Esses códigos a se completar estavam todos no código base fornecido [1]. Além disso, era necessário completar também o código da função *evaluate()* da classe *Simulation*, função essa a ser otimizada para o problema do robô seguidor de linha.

A análise de vários pontos dos algoritmos descritos acima terão uma breve descrição em alto nível da sua implementação a seguir.

Primeiramente, foi criada uma lista de partículas para a classe principal *ParticleSwarmOptimization*, um contador referente a qual partícula dessa lista o código estará se referindo em uma determinada iteração, além de variáveis para guardar o melhor valor (referente a função qualidade a ser otimizada) encontrado até aquela iteração dentre todas as partículas e qual a posição desse melhor valor.

Essa lista de partículas é formada por objetos da classe *Particle*, e cada partícula contém sua posição, sua velocidade, além da melhor posição referente ao valor calculado da função de qualidade e qual esse melhor valor alcançado pela partícula. A posição e a velocidade inicial de cada partícula é escolhida uniformemente aleatória, respeitando os limites estabelecidos por quem chama o construtor dessa classe.

As funções *get\_best\_position()* e *get\_best\_value()* retornam, respectivamente, a posição referente ao melhor valor encontrado na função qualidade e qual esse melhor valor dentre todas as partidas e todas as iterações. Já a função *get\_position\_to\_evaluate()* se utiliza do contador citado anteriormente para retornar o elemento correto na lista de partículas, ou seja, a próxima partícula a ser analisada.

Por fim, as funções *advance\_generation()* e *notify\_evaluation()* foram implementadas conforme apresentado nos pseudo códigos da seção Introdução, com alguns detalhes como utilizar o contador referente a qual partícula da lista de partículas o código está se referindo na iteração em questão em *notify\_evaluation()*. Além disso, para o caso do robô seguidor de linha, a função *evaluate()* da classe *Simulation* foi implementada conforme o apresentado na Seção 2 do roteiro do laboratório [1], com  $e_k = 1$  para o caso de não detecção de linha (booleana *detection* sendo *False*) e  $\omega = 0.5$ .

### III. RESULTADOS E CONCLUSÕES

A otimização foi executada para duas funções de qualidade: uma função matemática  $f(x) = -(x(0) - 1)^2 + (x(1) - 2)^2 + (x(2) - 3)^2$  e a função *evaluate()* do robô seguidor de linha. Os resultados das otimizações obtidas após a execução da implementação do algoritmo descrito acima foram apresentados nas Figuras de 1 a 6 para a função matemática, e de ?? a ?? para o robô seguidor de linha.

#### A. Função Esfera Transladada

A partir da Figura 2, foi possível notar o correto funcionamento da implementação do algoritmo PSO, uma vez que os parâmetros  $x(0)$ ,  $x(1)$  e  $x(2)$  convergiram corretamente para os valores 1, 2 e 3, como era esperado. Além disso, as Figuras 5 e 6 comprovaram a convergência da função para 0, valor máximo também já esperado.

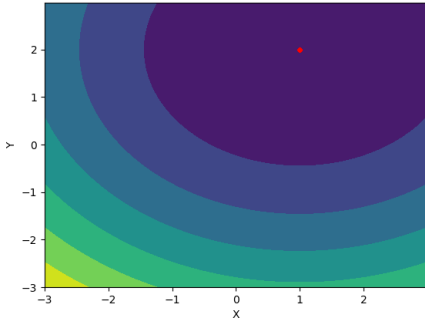


Figura 1. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

#### B. Função Ackley

#### C. Função Schaffer N° 2

#### D. Função Rastrigin (2D)

Tendo em vista o correto funcionamento do algoritmo PSO dado o apresentado na subseção anterior e o comportamento coerente realizado pelo robô apresentado na Figura ??, pode-se concluir que os resultados apresentados na Figura ?? são parâmetros satisfatórios do problema, chegando a convergência maximizada da Figura ??.

Esses valores de parâmetros foram encontrados após cerca de 3000 iterações, levando a um valor de função qualidade máximo de 572.33; parâmetros esses: a

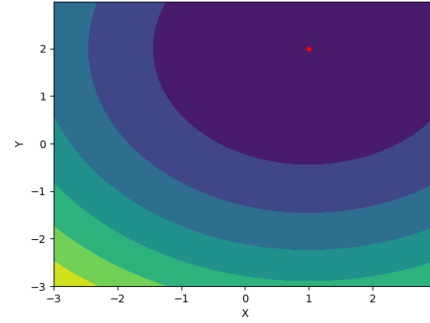


Figura 2. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

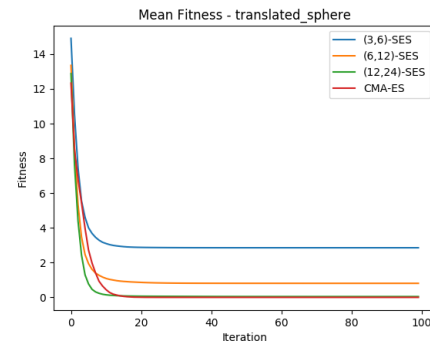


Figura 3. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

velocidade linear do robô (0.72) e os ganhos proporcional, integrativo e derivativo (131.73, 624.82, 15.37, respectivamente).

Tendo em vista o que foi apresentado, pode-se notar, por fim, que esse algoritmo realmente se demonstrou eficaz em encontrar parâmetros otimizados para uma determinada função de custo e um conjunto de possíveis soluções iniciais.

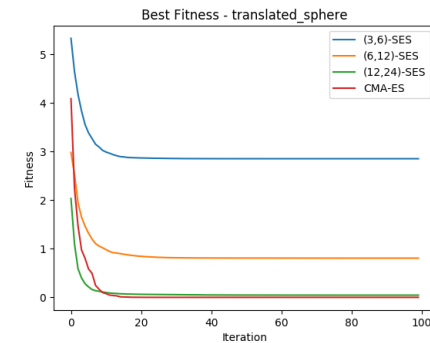


Figura 4. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

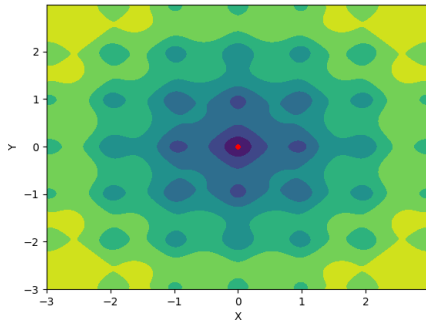


Figura 5. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

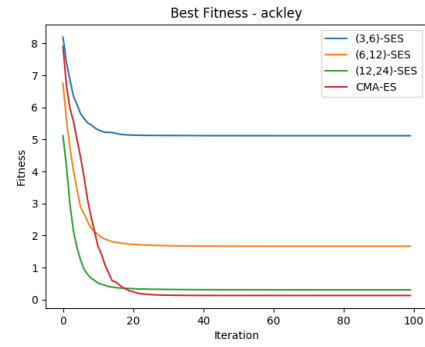


Figura 8. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

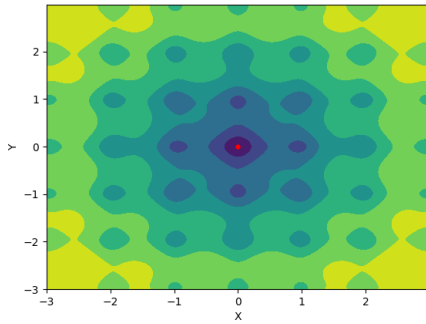


Figura 6. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

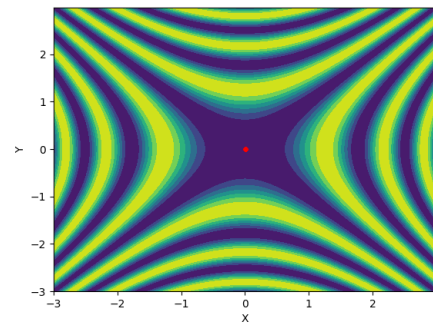


Figura 9. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

## REFERÊNCIAS

- [1] M. Maximo, “Roteiro: Laboratório 4 - Otimização com Métodos Baseados em População”. Instituto Tecnológico de Aeronáutica, Departamento de Computação. CT-213, 2019.

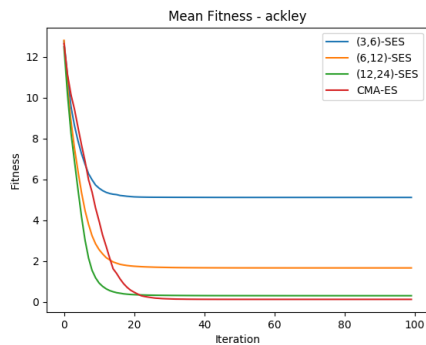


Figura 7. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

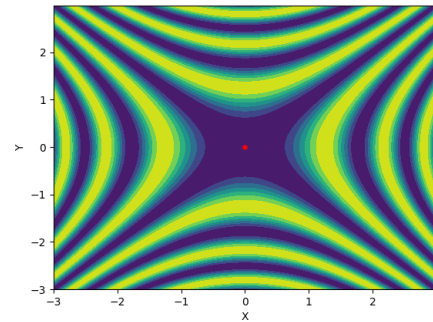


Figura 10. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

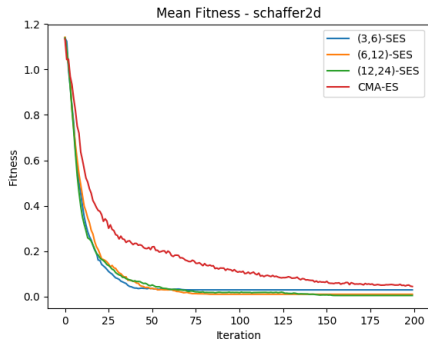


Figura 11. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

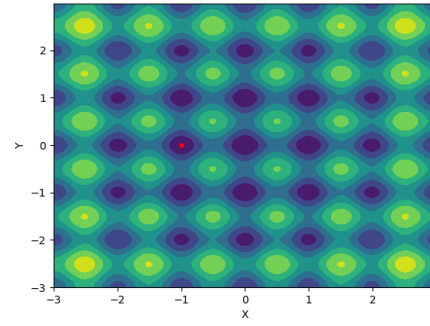


Figura 14. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

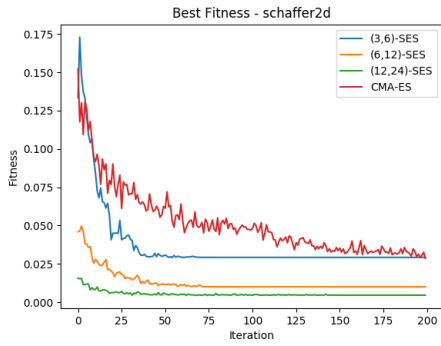


Figura 12. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

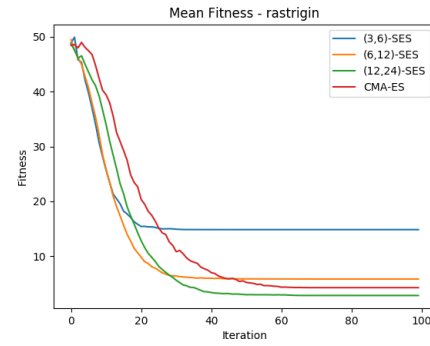


Figura 15. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

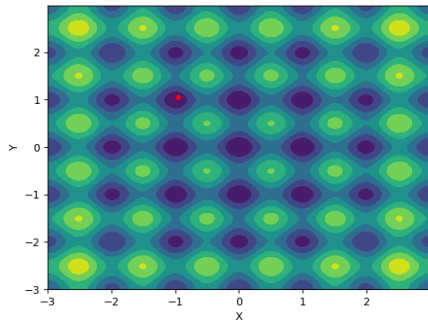


Figura 13. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

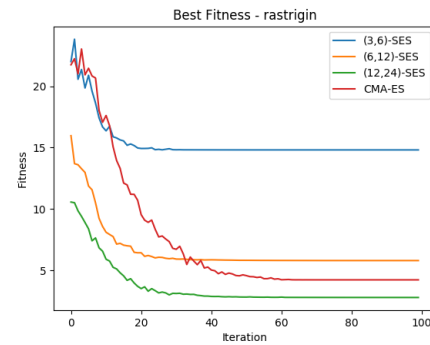


Figura 16. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.