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 fuções 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 chamade 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 algortimos 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 utilizando uma simples estratégia evolutiva (SES, do inglês *Simple Evolution Strategy*), era necessário preencher a função *tell()* da classe *SimpleEvolutionStrategy*. Recebendo os valores de *fitness* encontrados na população anterior, essa função era a responsável por atualizar a média e a matriz de covariância utilizando os melhores avaliados na antiga população e também evoluir a própria população a cada iteração. Essa função a se completar estava no código base fornecido [1].

Além disso, era necessário comparar os desempenhos desse algoritmo SES com o algoritmo CMA-ES, já fornecido no código base, aplicando-os na otimização de quatro funções, a se saber: esfera transladada, e as fuções de Ackley, Schaffer 2 e Rastrigin 2D.

A análise de vários pontos do algoritmo descrito acima terá uma breve descrição em alto nível da sua implementação a seguir.

Primeiramente, foi necessário ordenar a população atual tendo em vista os valores de *fitness* associados a ela recebido por parâmetro. Isso foi feito utilizando a função *argsort()* da biblioteca *Numpy*, conforme sugerido pelo roteiro do labotatório.

Após isso, as *mu* melhores amostras dessa população foram separadas para ajudar no cálculo de sua matriz de covariância e, posteriormente, realizar o cálculo da nova média das melhores amostras dessa geração em questão. Essa matriz de covariância foi feita a partir da multiplicação de uma determinada matriz auxiliar pela sua transposta, sendo essa matriz auxiliar a diferença entre essas melhores amostras e a média encontrada na população anterior.

Dessa forma, tendo em posse a nova matriz de covariância e a nova média das melhores amostras da população anterior, é possível gerar uma nova população, evoluindo para a próxima geração de possíveis candidatas a solução.

Por fim,

a fim de realizar o *benchmark* através de simulações de Monte Carlo para comparar os desempenhos dessa estratégia evolutiva simples e do CMA-ES, foi alterado novamente os valores das variáveis *algorithm* e *function*, dessa vez no arquivo *benchmark_evolution_strategy.py* do código base, gerando dessa vez os gráficos comparativos de rendimento para diferentes situações de SES e CMA-ES. Esses gráficos foram apresentados nas Figuras 3, 4, 7, 8, 11, 12, 15 e 16.

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.

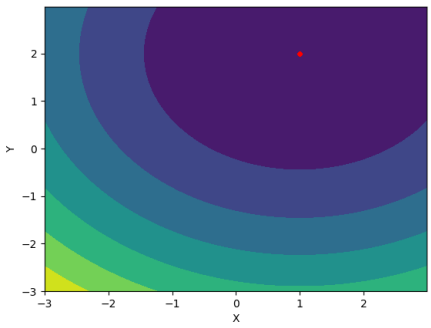

Figura 1. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

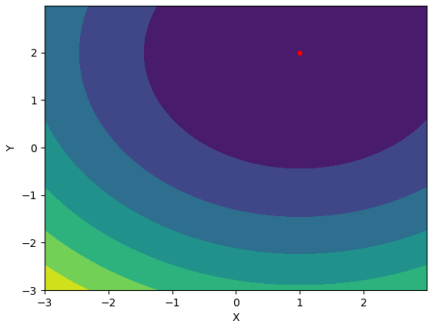

Figura 2. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

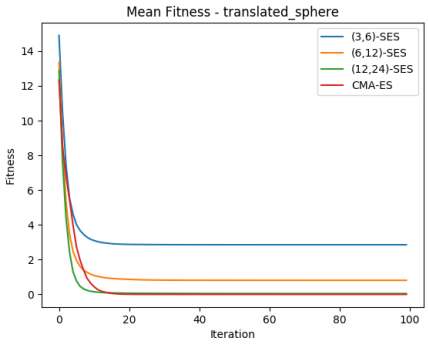

Figura 3. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

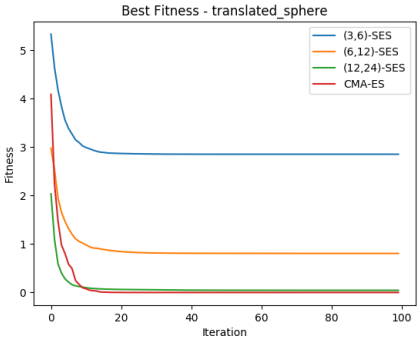

Figura 4. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

B. Função Ackley

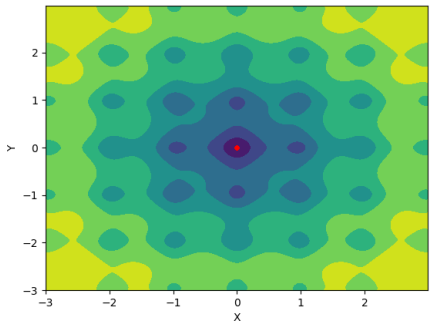

Figura 5. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

C. Função Schaffer Nº 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

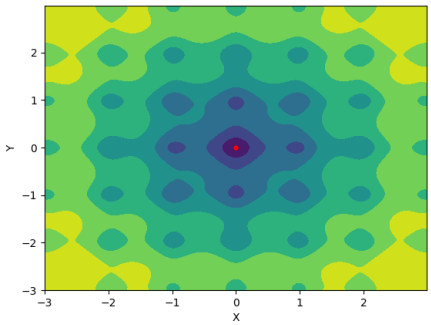

Figura 6. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

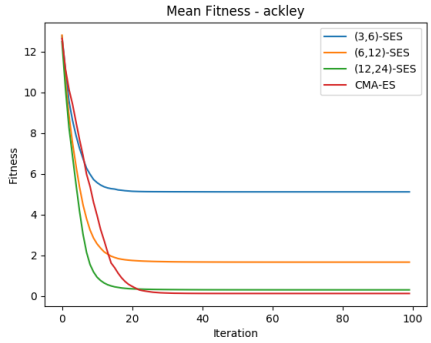

Figura 7. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

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 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.

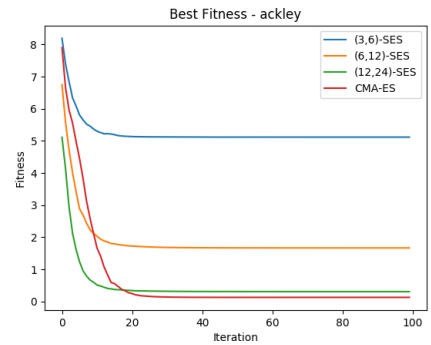

Figura 8. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

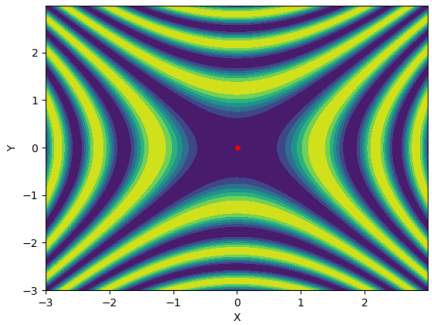

Figura 9. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

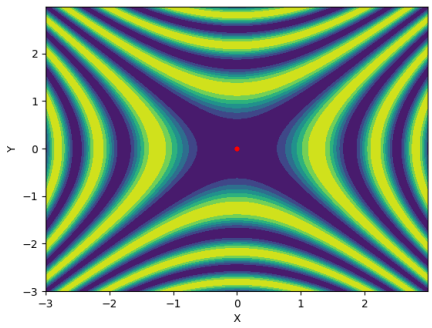

Figura 10. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

Referências

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

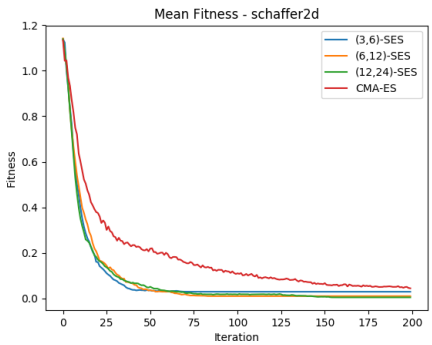

Figura 11. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

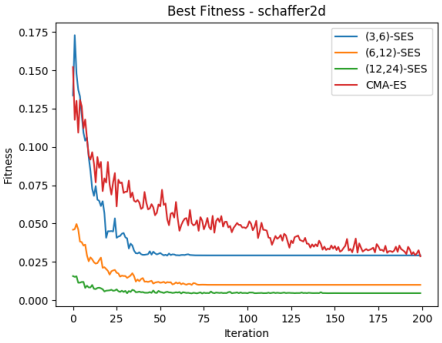

Figura 12. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

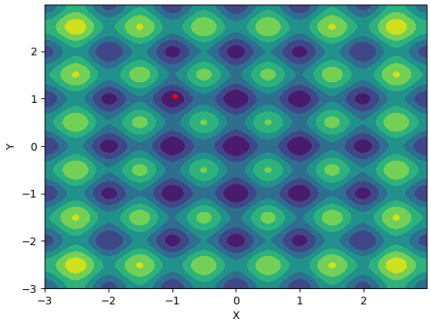

Figura 13. Convergência do valor da função qualidade com o passar das iterações para o caso da função matemática.

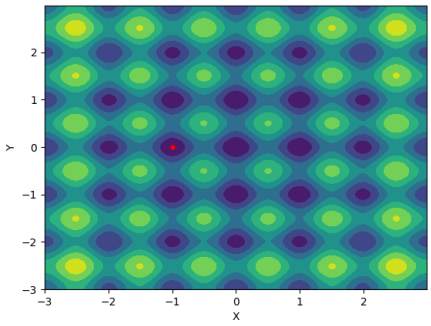

Figura 14. Convergência do melhor valor da função qualidade encontrado com o passar das iterações para o caso da função matemática.

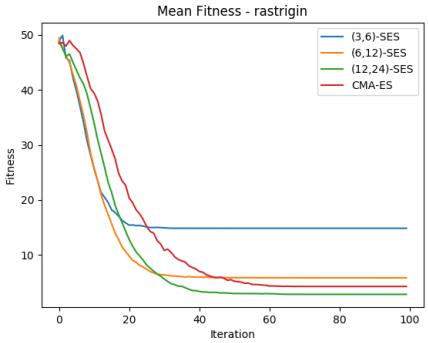

Figura 15. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.

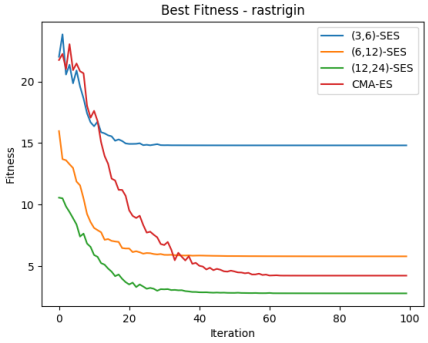

Figura 16. Valores convergidos dos parâmetros para maximização da função para o caso da função matemática.