
درس رایانش تکاملی مبتنی بر جمعیت

تمرین دوم

# یک الگوریتم استراتژی تکاملی ساده بر روی توابع ارزیابی معیار

نام دانشجو:

نوید حاجی زاده

۴۰۱۲۳۶۶۰۵۷

نام استاد درس:

دکتر مجتبی روحانی

پاییز ۱۴۰۲

## چکیده

در این گزارش قصد داریم از الگوریتم استراتژی تکاملی برای حل مسائل بهینه سازی استفاده کنیم. برای به حداقل رساندن مقادیر fitness از دو تابع f۱ و f۵ بهره برده ایم. الگوریتم استراتژی تکاملی برای ایجاد جمعیتی از راه حل‌های مختلف از جهش استفاده کرده و این جهش‌ها با استفاده از پارامتر سیگما تطبیق پیدا می‌کند. حال با تغییر در اندازه جمعیت (n)، اندازه گام (سیگما) ، نرخ موفقیت و ارزش fitness در طول نسل‌های مختلف می‌توان این راه حل‌ها را ارزیابی کرد که آیا موفقیت آمیز بوده اند یا خیر. در نهایت نتایج را به صورت نمودارهای لگاریتمی رسم کرده ایم و در پیوست نیز کد پایتون آورده شده است.

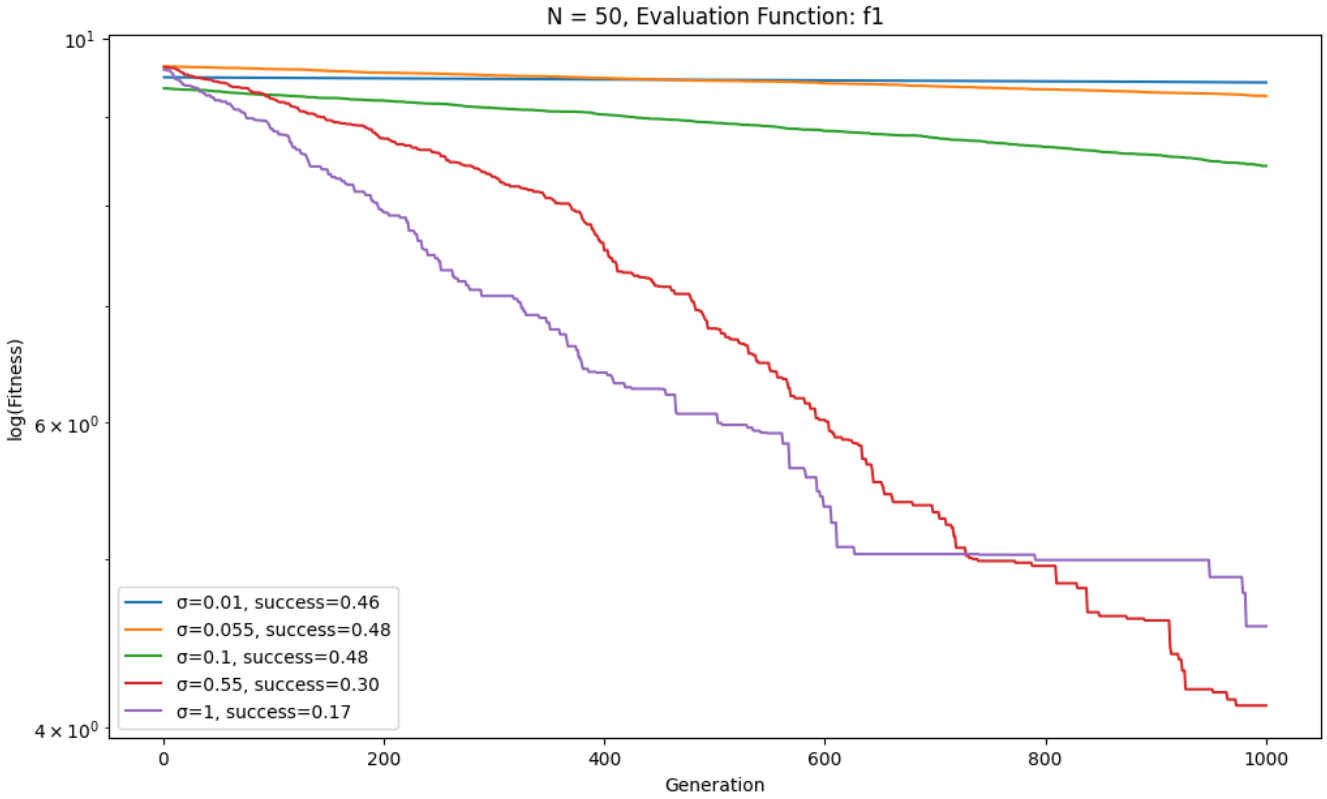
## شرح مسئله :

الگوریتم استراتژی تکاملی یک روش بهینه‌سازی مبتنی بر تکامل است که به شبیه‌سازی فرآیندهای تکاملی طبیعی مانند انتخاب طبیعی، تنوع ژنتیکی، تلاش برای تطابق به محیط و تغییرات تصادفی تکیه می‌کند. در این الگوریتم، یک جمعیت از افراد تولید می‌شود و هر جمعیت در آن یک نسخه از راه حل مسئله مورد نظر را نمایش می‌دهد. این افراد از طریق عملیات انتخاب، جهش و تلاش برای تطابق به محیط بهبود می‌یابند. طریق بهبود به این صورت است که جواب یا همان خروجی ارزیابی شده و اصطلاحا برازش یا fitness آن حساب گشته سپس اگر برازش مقدار جدید از مقدار قبلی بیشتر بود جایگزین مقدار قبلی می‌شود در غیر این صورت مقدار قبلی همچنان باقی می‌ماند. الگوریتم به صورت مراحل متعدد اجرا می‌شود و به تدریج بهینه‌ترین راه‌حل را پیدا می‌کند.

## شرح نتایج :

باید به این نکته توجه داشت که چون جهش‌ها تصادفی هستند نتایج هر اجرا با اجرای دیگر کمی متفاوت است ولی حالت کلی قابل تحلیل می‌باشد. همچنین پارامترهای الگوریتم استراتژی تکاملی طبق صورت مسئله تنظیم شده‌اند. به طور کلی نمودار زیر تغییرات لگاریتم برازندگی در طی نسل را نشان می‌دهد.

همانطور که در شکل زیر مشاهده می‌شود، بهترین جواب مربوط به منحنی قرمز رنگ است که نشان می‌دهد مقدار سیگمای ۰.۵۵ برای بهینه سازی تابع ارزیابی $f_1$ با مقدار $N= ۵۰$ مناسب ترین است زیرا سریع‌تر از مقادیر سیگمای دیگر به جواب بهینه رسیده ایم در واقع می‌توان گفت کمترین میزان لگاریتم برازندگی را در این حالت داریم.




با مشاهده هر کدام از نمودارها که در خروجی کد ماست و همچنین بعد از بررسی کامل تاثیر مقادیر مختلف جهش(σ) به این نتیجه خواهیم رسید که تاثیر مقدار سیگما ۰.۰۱ =σ بر روی نرخ موفقیت (success) از بقیه بیشتر است. بعد از آن به ترتیب ۰.۰۵۵و ۰.۱و ۰.۵۵و در نهایت ۱ خواهد بود.

## پیوست :

در این بخش کدهای مربوط به این الگوریتم آورده شده است.

```
from typing import Callable, List
import numpy as np
import pandas as pd
import json
import seaborn as sns
import matplotlib.pyplot as plt
```

```python
def f1(x: np.ndarray) -> float:
    return np.sum(x ** 2)

def f5(x: np.ndarray) -> float:
    term1 = 100 * (x[1:] - x[:-1] ** 2) ** 2
    term2 = (x[:-1] - 1) ** 2
    fitness = np.sum(term1 + term2)
    return fitness

def run_es(n: int, sigma: float, eval_func: Callable[[np.ndarray], float], low=-30,
high=30, n_func_evals=1000):
    x = np.random.uniform(low, high, n)
    initial_x = x
    z = np.random.randn(n)
    success_count = 0
    fitnesses = [eval_func(x)]

    for _ in range(n_func_evals - 1):
        parent_fitness = fitnesses[-1]
        child_x = x + sigma * z
        child_x = correct_range(child_x, low, high)
        child_fitness = eval_func(child_x)

        if child_fitness < parent_fitness:
            x = child_x
            fitnesses.append(child_fitness)
            success_count += 1
        else:
            fitnesses.append(parent_fitness)
        z = np.random.randn(n)

    return initial_x, x, success_count / n_func_evals, fitnesses

def correct_range(x: np.ndarray, low, high):
    return np.clip(x, low, high)

def plot_results(eval_func_name, ns, sigmas, success_rates, fitnesses):
    for i, n in enumerate(ns):
        plt.figure(figsize=(12, 7))
        plt.title(f"N = {n}, Evaluation Function: {eval_func_name}")
        plt.xlabel("Generation")
        plt.ylabel("log(Fitness)")
```

```
        for j, sigma in enumerate(sigmas):
            label = f"σ={sigma}, success={success_rates[i][j]:.2f}"
            plt.plot(range(1, len(fitnesses[i][j]) + 1), np.log(fitnesses[i][j]),
label=label)

        plt.yscale("log")
        plt.legend()
        plt.show()

def main():
    evaluation_functions = [f1, f5]
    ns = [2, 5, 10, 50]
    sigmas = [0.01, 0.055, 0.1, 0.55, 1]

    data = {}
    for eval_func in evaluation_functions:
        success_rates = []
        models_fitnesses = []

        for n in ns:
            success_rates_n = []
            models_fitnesses_n = []

            for sigma in sigmas:
                initial_x, best_x, success_rate, fitnesses = run_es(n, sigma,
eval_func)

                models_fitnesses_n.append(fitnesses)
                success_rates_n.append(success_rate)

            models_fitnesses.append(models_fitnesses_n)
            success_rates.append(success_rates_n)

        plot_results(eval_func.__name__, ns, sigmas, success_rates, models_fitnesses)

if __name__ == "__main__":
    main()
```