Homework2

110550126 □□□

Q1. Discussion on ES and Steady-State GAs as Extremes of Population Size and Offspring Creation

Evolution Strategies (ES)

- **Population Size:** ES generally uses **large populations**. This means many individuals explore different areas of the search space, reducing the risk of getting stuck in local optima.
- **Offspring Creation:** In each generation, ES generates a **large number of offspring** (often more than the parent population), allowing extensive exploration of the solution space.
- **Selection:** ES often uses **μ+λ** or **μ,λ** selection strategies, where offspring and parents compete for survival or only offspring are selected, respectively. This encourages diversity and broad search, helping the algorithm adapt quickly.

Steady-State Genetic Algorithms (SSGA)

- **Population Size:** SSGA maintains a **small, stable population** size. Only a few individuals are replaced at a time, making changes to the population slow and incremental.
- **Offspring Creation:** SSGA produces **one or two offspring** per iteration, leading to gradual change that focuses on fine-tuning rather than broad exploration.
- **Selection:** SSGA replaces weaker individuals with better offspring, so the population evolves slowly but steadily toward higher fitness.

Key Difference: Exploration vs. Exploitation

- **ES** aims for **exploration** by maintaining a large population and creating many offspring, seeking global optimization.
- **SSGA** leans toward **exploitation**, refining the existing population gradually to find a good local solution.

In essence, **ES** covers a broad search space with many offspring, while **SSGA** makes small, steady changes to a smaller population, focusing on fine-tuning.

Q2. feature frequency

Problem

Given:

- A population of (\mu) individuals, each represented as a bit-string of length (L).
- Let the frequency of allele 1 be 0.25 at position i, that is, 25% of all individuals contains a 1, and 75% a 0 at the ith position on the chromosome.

Determine how the allele frequency changes after performing (k) crossover operations with:

1. One-point crossover.
2. Uniform crossover.

One-Point Crossover

In one-point crossover, a random crossover point is selected, and the segments to the right of that point in two parent bit-strings are swapped.

However, after a crossover the allele frequency at position (i) will remain at 0.25, since the origin of both is 0.25. So after k crossover the allele frequency at position (i) will remain at 0.25

Uniform Crossover

In uniform crossover, each gene (bit) position is independently swapped between the two parents with a certain probability (often 0.5).

However, after a crossover the allele frequency at position (i) will remain at 0.25, since the origin of both is 0.25. So after k crossover the allele frequency at position (i) will remain at 0.25

Summary

For both one-point and uniform crossover, the allele frequency at position (i) will remain at 0.25 after (k) crossovers. This is because crossover operations are recombination techniques that do not inherently change allele frequencies; they only rearrange the genetic material among individuals in the population.

Q3. Tables of (1,1)-ES and (1+1)-ES using correlated Gaussian mutation

(1, 1)-ES Results

Run #	**σ = 0.01**	**σ = 0.1**	**σ = 1.0**
1	10000000	10000000	10000000
2	10000000	10000000	10000000
3	10000000	10000000	10000000
4	10000000	10000000	10000000
5	10000000	10000000	10000000
6	10000000	10000000	10000000
7	10000000	10000000	10000000
8	10000000	10000000	10000000
9	10000000	10000000	10000000
10	10000000	10000000	10000000

Run #	σ = 0.01	σ = 0.1	σ = 1.0
mean	10000000	10000000	10000000

(1 + 1)-ES Results

Run #	σ = 0.01	σ = 0.1	σ = 1.0
1	804	35789	10000000
2	908	506074	10000000
3	790	21242	10000000
4	794	416679	10000000
5	835	68179	10000000
6	792	26027	10000000
7	814	1627	10000000
8	904	439967	10000000
9	841	318044	10000000
10	756	359279	10000000
mean	823.8	218660.8	10000000

Q4. Observe the running processes of problem 3

problem

Compare and contrast the results you obtained in problem 3 and discuss what you think about the difference between (1 1)-ES and (1+1)-ES.

step size (\sigma)

- **(1,1)-ES Results**:
    - Across all values of (\sigma), the (1,1)-ES method consistently reached the maximum number of iterations (10,000,000) without meeting the objective threshold of 0.005.
    - The lack of improvement regardless of step size suggests that this strategy lacks an efficient selection mechanism to focus on better solutions.
- **(1+1)-ES Results**:
    - The (1+1)-ES strategy demonstrated variability in results based on the step size:
        - For **(\sigma = 0.01)** and **(\sigma = 0.1)**, the strategy converged in fewer generations, with average generation counts of 823.8 and 218,660.8, respectively.

        - With **(\sigma = 1.0)**, however, the (1+1)-ES also reached the maximum number of iterations, indicating excessive variance and ineffective convergence with large mutation sizes.

Strategy Type

- **(1,1)-ES**:
    - This strategy replaces the parent with the offspring regardless of fitness improvement, leading to a more exploratory search that lacks focus on fitter regions. Consequently, it behaves as a random walk, often failing to converge in constrained runs.
- **(1+1)-ES**:
    - The (1+1)-ES employs a selective replacement mechanism, where the offspring replaces the parent only if it has a better fitness. This selection pressure allows the algorithm to refine the solution around promising areas and promotes faster convergence.
    - The strategy is more effective in balancing exploration and exploitation, particularly with smaller (\sigma) values, where controlled mutations lead to finer adjustments.

Summary of Observations

- **Efficiency of Selective Replacement**: The (1+1)-ES's selective replacement allows it to maintain beneficial traits and converge more effectively, especially at smaller (\sigma) values. In contrast, the (1,1)-ES's random replacement hinders convergence.
- **Influence of Step Size**
- **Influence of Step Size**: Smaller step sizes (e.g., (\sigma = 0.01)) enhance the (1+1)-ES's ability to refine solutions, while larger steps increase the risk of excessive divergence in both strategies.

In conclusion, the (1+1)-ES provides a more efficient approach for correlated Gaussian mutation optimization, as its selective mechanism enables it to focus on fit individuals, whereas the (1,1)-ES is less suited for tasks requiring rapid convergence due to its lack of selection pressure.

Q5. Tables of (1,1)-ES and (1+1)-ES using uncorrelated Gaussian mutation

(1, 1)-ES Results

Run #	σ = 0.01	σ = 0.1	σ = 1.0
1	10000000	10000000	10000000
2	10000000	10000000	10000000
3	10000000	10000000	10000000
4	10000000	10000000	10000000

Run #	σ = 0.01	σ = 0.1	σ = 1.0
5	10000000	10000000	10000000
6	10000000	10000000	10000000
7	10000000	10000000	10000000
8	10000000	10000000	10000000
9	10000000	10000000	10000000
10	10000000	10000000	10000000
mean	10000000	10000000	10000000

(1 + 1)-ES Results

Run #	σ = 0.01	σ = 0.1	σ = 1.0
1	7411	4205	4373
2	1768	3848	7586
3	12750	3568	759
4	7636	4403	37801
5	2093	6197	3789
6	7342	4073	4612
7	9265	2849	3547
8	5954	5049	4957
9	3837	6764	2208
10	4027	13893	5214
mean	6208.3	5484.9	7484.6

Q6. Comparison of Results from Problems 3 and 5

(1, 1)-ES with Correlated vs. Uncorrelated Mutation:

- **Correlated and Uncorrelated Gaussian Mutations**: Both types of mutation failed to converge to the target within the 10 million iterations allowed for the **(1, 1)-ES** strategy, regardless of the step size. This suggests that neither type of mutation was effective enough in this context to adapt the search process and reach the desired solution.
- **Lack of Adaptability in (1, 1)-ES**: The absence of self-adaptive mechanisms in **(1, 1)-ES** reflects a challenge in exploring the solution space. Without the ability to adjust the mutation step dynamically, the algorithm remains highly sensitive to the starting point and initial step size, making it difficult to find an optimal solution. This points to a

limitation of the (1, 1)-ES method, as it cannot efficiently explore the space or adapt to the complexity of the optimization task.

(1 + 1)-ES with Correlated Mutation:

- **Step Size Dependency**: In the case of **correlated mutation** within **(1 + 1)-ES**, the performance was heavily influenced by the step size. Smaller step sizes ($\sigma = 0.01$ and 0.1) performed better, with fewer iterations required to converge compared to larger step sizes ($\sigma = 1.0$), where convergence was not achieved within the allowed iterations.
- **Failure with Larger Step Sizes**: Even with **correlated mutation**, large step sizes ($\sigma = 1.0$) failed to produce a result within the maximum iterations. This suggests that the mutation mechanism, while effective for small step sizes, struggled to handle larger step sizes, pointing to an insufficiency in adjusting the mutation step for different stages of the optimization process.

(1 + 1)-ES with Uncorrelated Mutation:

- **Self-Adaptation in Uncorrelated Mutation**: **Uncorrelated Gaussian mutation**, with its self-adaptive step size mechanism, showed significant improvement over **correlated mutation**. Particularly with smaller and moderate step sizes, the self-adaptive feature allowed the algorithm to converge much faster and with fewer iterations.
- **Better Performance at Smaller Step Sizes**: The self-adaptive mechanism allowed the **(1 + 1)-ES** strategy to fine-tune its mutation step dynamically. This helped the algorithm adapt to the fitness landscape more efficiently, leading to a noticeable improvement in convergence speed, especially when compared to the results from the correlated mutation.
- **Degraded Performance with Larger Step Sizes**: While the self-adaptive uncorrelated mutation showed better performance overall, its effectiveness decreased when using a large step size ($\sigma = 1.0$). This still represented an improvement over correlated mutation, though it highlighted that even self-adaptive mechanisms struggle with overly large mutation steps in certain scenarios.

Self-Adaptation:

- **Self-Adaptation's Role**: The **self-adaptation** seen in **uncorrelated Gaussian mutation** was key to its improved performance in **(1 + 1)-ES**. It allowed the algorithm to adjust the mutation step size dynamically based on the progress in the optimization process. This adaptability helps prevent overshooting the target when large steps are used and allows for finer exploration with smaller steps, helping the algorithm avoid getting stuck in local optima.
- **Advantages of Self-Adaptation**: The ability to adjust the mutation strength as needed enabled the algorithm to strike a balance between exploration (searching broadly across the space) and exploitation (refining the solution). This is particularly useful in complex

landscapes where an initial fixed step size may not work well for the entire optimization process.

- **Challenges Without Self-Adaptation**: In contrast, the **(1, 1)-ES** with **correlated mutation** struggled because the algorithm couldn't adjust the step size dynamically. This lack of adaptability made it difficult to effectively balance exploration and exploitation, leading to failure in finding an optimal solution.

Key Parameters:

- **n = 10**: The dimensionality of the optimization problem, determining the size of the solution vector.
- **max_iterations = 10,000,000**: The upper limit on the number of iterations for the algorithm to reach the target solution.
- **target = 0.005**: The desired value of the objective function (f(x)) that the algorithm aims to achieve.
- **τ (tau) = 1 / np.sqrt(2 * np.sqrt(n))**: A parameter that controls the step size adaptation, ensuring the evolution process is stable and diverse based on the problem's dimensionality.
- **τ' (tau_prime) = 1 / np.sqrt(2 * n)**: Another adaptive parameter for adjusting mutation step sizes globally, based on the dimension of the problem.
- **ε_0 (epsilon) = 1e-6**: A small value that helps prevent numerical issues by avoiding overly small mutation steps and ensuring stable convergence.

Conclusion:

Self-adaptation, particularly in the case of **uncorrelated Gaussian mutation** within the **(1 + 1)-ES** strategy, provides a more effective and flexible search mechanism than the static mutation step sizes used in **(1, 1)-ES**. By dynamically adjusting the step size, the algorithm is able to explore the solution space more efficiently and avoid issues like overshooting or stagnation, ultimately resulting in faster convergence and better performance.

Q7. Tables of (1,1)-ES and (1+1)-ES using 1/5-rule and Gaussian mutation

(1, 1)-ES Results

Run #	σ = 0.01	σ = 0.1	σ = 1.0
1	10000000	10000000	10000000
2	10000000	10000000	10000000
3	10000000	10000000	10000000
4	10000000	10000000	10000000
5	10000000	10000000	10000000

Run #	σ = 0.01	σ = 0.1	σ = 1.0
6	10000000	10000000	10000000
7	10000000	10000000	10000000
8	10000000	10000000	10000000
9	10000000	10000000	10000000
10	10000000	10000000	10000000
mean	10000000	10000000	10000000

(1 + 1)-ES Results

Run #	σ = 0.01	σ = 0.1	σ = 1.0
0	1556	661	381
1	871	322	1026
2	505	733	627
3	670	498	619
4	644	898	763
5	362	511	272
6	2217	609	651
7	574	1940	488
8	851	483	334
9	792	1183	681
mean	904.2	783.8	584.2

Q8. Comparison of (1,1)-ES and (1+1)-ES Results with Different Mutation Strategies

Summary of Observations

1. **Correlated Gaussian Mutation**:
    - For **(1,1)-ES**, none of the tests converged within the limit of 10 million iterations, regardless of the mutation rate (σ). This suggests that the mutation strategy struggled to effectively find solutions.
    - With **(1+1)-ES**, results varied: small mutation rates (like $\sigma = 0.01$ and $\sigma = 0.1$) generally allowed faster convergence, while the largest rate ($\sigma = 1.0$) didn't help the algorithm reach a solution in most cases.
2. **Uncorrelated Gaussian Mutation**:

- **(1,1)-ES** showed similar issues to correlated mutation, with all runs maxing out at 10 million iterations and no successful convergence.
- **(1+1)-ES**, however, performed much better. The majority of tests converged well before reaching 10 million iterations, especially with smaller mutation rates ($\sigma = 0.01$) and ($\sigma = 0.1$)). The larger mutation rate (($\sigma = 1.0$)) had more mixed outcomes but still showed some promise.

3. **1/5-Rule with Gaussian Mutation**:

   - Again, **(1,1)-ES** couldn’t reach a solution within 10 million iterations.
   - For **(1+1)-ES**, though, applying the 1/5-rule led to a noticeable improvement. The algorithm converged significantly faster with all mutation rates, reducing the average number of iterations needed to find solutions compared to the previous methods.

Key Takeaways and Comparisons

- **Why the 1/5-Rule Helped**: The 1/5-rule works by adjusting the mutation step size (how far solutions can "jump" in each step) based on the algorithm’s success rate. If progress slows down, the step size increases to help explore the search space more. When near a solution, it decreases to fine-tune the final steps. This adaptability was a clear advantage, especially with higher mutation rates that previously struggled to converge.

- **Comparing Mutation Strategies**:

  - The **(1+1)-ES approach** consistently outperformed (1,1)-ES because it’s designed to keep the best solution at each step, making it more likely to progress steadily toward a solution.
  - **Correlated vs. Uncorrelated Mutation**: Uncorrelated mutation, where each dimension can change independently, was more effective than correlated mutation. This freedom helps the algorithm better navigate the search space.
  - **1/5-Rule’s Advantage**: By allowing self-adjustment, the 1/5-rule enabled the algorithm to better balance exploring new solutions with focusing on promising ones. It excelled particularly in cases where static mutation rates led to either slow progress or overshooting a solution.

Conclusion

The 1/5-rule is a powerful tool for self-adjusting mutation rates based on how successful the algorithm is at each step. By adapting to the problem dynamically, it can effectively guide the search process. This strategy proved especially useful for challenging or noisy problems, where other methods couldn’t converge within the maximum allowed steps. The 1/5-rule’s success here highlights its value in making mutation strategies more robust and adaptable.