آرمان نژادسلیمانی – مائده رهنمافر

1) Roulette Wheel Selection :روش استفاده‌شده

.در این روش، احتمال انتخاب هر تور به‌صورت تناسبی با مقدار فیتنس آن است

فیتنس به‌صورت معکوس فاصله کل تور محاسبه می‌شود :

$$\text{Fitness} = \frac{1}{\text{Distance}}$$

والدین با توجه به احتمال تناسب انتخاب می‌شوند.

2) (ترکیب والدین) Crossover

Partial Mapped Crossover (PMX) :روش استفاده‌شده

دو نقطه تصادفی در طول تور انتخاب می‌شود.

ژن‌های بین این دو نقطه از والد اول به فرزند کپی می‌شوند.

بقیه ژن‌ها از والد دوم تکمیل می‌شوند به‌گونه‌ای که ترتیب اصلی والد دوم حفظ شود.

3)
(جهش) Mutation:

Swap Mutation :روش استفاده‌شده

دو موقعیت تصادفی در تور انتخاب می‌شود

ژن‌های این دو موقعیت با هم جابه‌جا می‌شوند

4) (تولید نسل) Generation

فرزندان جدید جایگزین کل جمعیت می‌شوند

الگوریتم به تعداد نسل‌های مشخص (در اینجا 500) تکرار می‌شود

5) (محاسبه فیتنس) Fitness Calculation

روش استفاده‌شده:

فاصله تور با استفاده از فرمول فاصله اقلیدسی بین نقاط محاسبه می‌شود

$$\text{Distance} = \sqrt{^2(_1y_2 - y) + ^2(_1x_2 - x)}$$

تغییر روش (جابجایی) Swap Mutation به روش (جهش گاوسی) Gaussian Mutation

```python
    def mutate_gaussian(self, tour):
        if np.random.rand() < self.mutation_rate:
            idx = np.random.randint(len(tour))  # انتخاب یک شهر تصادفی
            point = np.array(self.points[tour[idx]])
            new_point = point + np.random.normal(loc=0.0, scale=0.1, size=2)  # تغییر با توزیع نرمال

            # جایگزینی مختصات جدید
            self.points[tour[idx]] = tuple(new_point)

    def run(self):
        for generation in range(self.generations):
            new_population = []
            for _ in range(self.population_size):
                parent_indices = self.select_parents()
                parent1 = self.population[parent_indices[0]]
                parent2 = self.population[parent_indices[1]]
                child = self.crossover(parent1, parent2)
                self.mutate_gaussian(child)
                new_population.append(child)

            self.population = new_population

        best_tour = min(self.population, key=self.calculate_distance)
        best_distance = self.calculate_distance(best_tour)
        return best_tour, best_distance

ga = GeneticAlgorithm(points)
best_tour, best_distance = ga.run()

# Print best tour
cities = iran_df['City'].tolist()
print("Best tour:", [cities[i] for i in best_tour])
print("Best distance:", best_distance)
```

```
Best tour: ['Arak', 'Kermanshah', 'Qom', 'Urmia', 'Zanjan', 'Bandar Abbas', 'Kerman', 'Tabriz', 'Gorgan', 'Sari', 'Yazd', 'Ahvaz', 'Mashhad', 'Birjand', 'Bojnurd', 'Sabzevar', 'Shiraz', 'Isfahan', 'Tehran', 'Karaj']
Best distance: 100.19415311717593
```

مقایسه نتایج :

روش (جابجایی) Swap Mutation :

در این روش، دو شهر در مسیر به‌صورت تصادفی جابجا می‌شوند

این روش توانسته مسیر کوتاه‌تری با فاصله **99.40 واحد** پیدا کند

دلیل عملکرد بهتر این است که جابجا کردن شهرها تأثیر مستقیمی روی کاهش طول مسیر دارد.

Gaussian Mutation جهش گاوسی :

در این روش، مختصات جغرافیایی یک شهر به‌صورت تصادفی و با تغییرات کوچک (جهش گاوسی) تغییر می‌کند.

این روش به فاصله **100.19واحد** رسیده که کمی بدتر از روش قبلی است.

دلیل این نتیجه این است که تغییر مختصات مختصری که جهش گاوسی ایجاد می‌کند، برای مسئله فروشنده دوره‌گرد که ترتیب شهرها اهمیت بیشتری دارد، چندان کارآمد نیست.

Swap Mutationبهتر عمل کرده است، چون این روش مستقیماً ترتیب شهرها را تغییر می‌دهد و سریع‌تر به مسیر کوتاه‌تر می‌رسد.

Gaussian Mutation برای مسائلی بهتر است که تغییرات کوچک در مختصات تأثیر زیادی روی نتیجه داشته باشند، اما در این مسئله، ترتیب شهرها اهمیت بیشتری دارد.