تمرین تحویلی سری چهارم

محمدرضا سالاری

سوال یک:

- برای Selection از روش Truncation Selection (انتخاب بریدگی) استفاده شده چرا که افراد جمعیت براساس شایستگی مرتب می‌شوند و فقط نیمی از بهترین‌ها انتخاب می‌شوند. در روش Truncation Selection افراد جمعیت براساس شایستگی مرتب می‌شوند سپس درصدی از بهترین افراد انتخاب می‌شوند و بقیه حذف می‌شوند.
- برای crossover از روش (Ordered Crossover (OX1 استفاده شده است. در این روش بخشی از ژن‌های والد اول به فرزند کپی می‌شود سپس باقی ژن‌ها براساس ترتیب حضورشان در والد دوم تکمیل می‌شوند. در مسائل مسیریابی بسیار متداول است، چون ترتیب شهرها را حفظ می‌کند و هیچ شهر تکراری در مسیر وجود ندارد.
- برای mutation این کد از Swap Mutation (جهش تبادلی) استفاده می‌کند. در این روش دو شهر به صورت تصادفی انتخاب شده و موقعیتشان با یکدیگر جابه‌جا می‌شود. این روش سریع است و به خوبی تنوع جمعیت را حفظ می‌کند و فقط دو موقعیت جابه‌جا می‌شوند، بنابراین مسیر به شدت تغییر نمی‌کند.
- برای generation روش Generational Replacement استفاده شده است. در این روش، پس از تولید نسل جدید، نسل قبلی کاملاً با نسل جدید جایگزین می‌شود. از آنجا که در هر نسل تعداد زیادی فرد جدید تولید می‌شود، این روش می‌تواند تنوع خوبی به جمعیت بدهد. با این روش، فقط افراد شایسته نسل جدید می‌سازند، و امکان بهبود سریع وجود دارد.
- برای محاسبه fitness در این کد، fitness به صورت معکوس طول مسیر محاسبه می‌شود. این روش یکی از رایج‌ترین روش‌ها در مسئله TSP است که در آن هرچه طول مسیر کوتاه‌تر باشد، شایستگی بیشتر است. این روش برای الگوریتم‌های ژنتیک در مسائل بهینه‌سازی مسیر بسیار موثر است.

سوال دو:

روشی به نام "Partially Mapped Crossover PMX" به این صورت عمل می‌کند:

1. انتخاب دو نقطه‌ی تصادفی در والدین.
2. بخش بین این دو نقطه در والدین به عنوان تکه‌های مبادله در نظر گرفته می‌شود.
3. جایگذاری این تکه‌ها در فرزند.
4. تکمیل بقیه ژن‌ها با استفاده از نگاشت‌های جزئی بین والدین.

مراحل دقیق‌تر

1. انتخاب دو نقطه‌ی تصادفی.
2. نگاشت تکه‌های بین این دو نقطه.
3. جایگذاری تکه‌ها در فرزند.
4. تکمیل بقیه ژن‌ها با استفاده از نگاشت‌ها.

```
def pmx_crossover(parent1, parent2):
    start, end = sorted(np.random.choice(len(parent1), 2, replace=False))
    child = [None] * len(parent1)

    # Copy the segment from parent1 to the child
    child[start:end+1] = parent1[start:end+1]

    # Mapping elements between parents
    mapping = {}
    for i in range(start, end+1):
        mapping[parent2[i]] = parent1[i]

    # Fill the rest of the child
    for i in range(len(child)):
        if child[i] is None:
            current_gene = parent2[i]
            while current_gene in mapping:
                current_gene = mapping[current_gene]
```

```
            child[i] = current_gene

    return np.array(child)

# Example usage:
parent1 = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
parent2 = np.array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])

child = pmx_crossover(parent1, parent2)
print("Parent 1:", parent1)
print("Parent 2:", parent2)
print("Child:", child)
```

در این کد، دو والد انتخاب می‌شوند و بخش‌های بین دو نقطه تصادفی در والدین نگاشت می‌شوند تا فرزند تولید شود. این روش می‌تواند تنوع بیشتری به جمعیت بدهد و به بهبود نتایج کمک کند.

تفاوت این دو روش در این است که:
PMX از نگاشت جزئی برای تکمیل فرزند استفاده می‌کند، در حالی که **OX1** از ترتیب والد دوم برای تکمیل ژن‌های فرزند استفاده می‌کند.
PMX بیشتر بر روی نگه‌داشتن تکه‌های مشخص ژن‌ها و نگاشت‌های آنها تمرکز دارد، در حالی که **OX1** بیشتر بر حفظ ترتیب ژن‌ها تأکید می‌کند.