به نام خدا

علی عطاءاللهی

پروژه دوم هوش مصنوعی

پاسخ سوالات تشریحی :

۱. جمعیت اولیه ی بسیار کم یا بسیار زیاد چه مشکلاتی را به وجود می آورند؟

جمعیت بسیار کم : تنوع پاسخ هایی که داریم کم خواهد بود و تنها موردی که باعث خواهد شد تنوع ایجاد شود mutation خواهد بود. به همین دلیل اگر نسل پاسخ هایمان در مسیر خوبی قرار نگیرند بسیار دیر به پاسخ خواهیم رسید.

جمعیت بسیار زیاد : زمان بدست آوردن پاسخ زیاد خواهد شد و میزان حافظه مصرفی افزایش می‌یابد.

۲. اگر تعداد جمعیت در هر دوره افزایش یابد، چه تاثیری روی دقت و سرعت الگوریتم می گذارد؟

بستگی به نحوه پیاده سازی آن دارد. اگر به‌صورت درستی پیاده سازی شود باعث می‌شود تنوع در نسل خود داشته باشیم و خروجی الگوریتم کراس اوور بهتر خواهد بود.

۳. تاثیر هر یک از عملیات های crossover و mutation را بیان و مقایسه کنید. آیا می‌توان فقط یکی از آنها را استفاده کرد؟

در الگوریتم mutation بیشتر تنوع در جواب های خود ایجاد می‌کنیم و باعث می‌شویم راه های جدیدی ایجاد بشوند. در الگوریتم crossover ولی بین پاسخ های خود سعی می‌کنیم یک برآیند مناسب بگیریم و بدین ترتیب راه های جدید ایجاد کنیم.

۴ . به نظر شما چه راهکارهایی برای سریعتر به جواب رسیدن در این مسئله ی خاص وجود دارد؟

اگر در ورودی ما ضرب و ۲ و ۵ وجود داشته باشد. بهتر است بخشی از نسل اولیه را اعدادی تماما از این دو عدد تشکیل بدهند. یا کار دیگری که می‌توانیم بکنیم این است که اگر فاصله ما با جواب زیاد است ، احتمال انتخاب ضرب را بیشتر بکنیم.

۵ . با وجود استفاده از این روش ها، باز هم ممکن است که کروموزوم ها پس از چند مرحله دیگر تغییر نکنند. دلیل این اتفاق و مشکلاتی که به وجود می آورد را شرح دهید. برای حل آن چه پیشنهادی می دهید؟)راه حلهای خود را امتحان کنید و بهترین آن ها را روی پروژه خود پیاده سازی کنید).

علت : الگوریتم را به خوبی پیاده سازی نکرده باشیم، به طوری که راه حل های جدید را ایجاد نکنیم یا به صورت درست با هم کراس اوور نکرده باشیم. علت دیگر این می‌تواند باشد که درصد خوبی از جواب های خوبمان را به نسل بعد نمی‌بریم در نتیجه هیچ وقت به سمت جواب درست حرکت نمی‌کنیم.

راه حل: استفاده بیشتر و بهتر از mutation و این که تعداد قابل قبولی از کروموزم ها را در هر نسل بررسی کنیم و درصد خوبی از آنها را به نسل بعدی ببریم. همچنین کراس اوور مناسب نیز تاثیر گذار خواهد بود(در پروژه این موارد بررسی و اعمال شده‌اند)

البته این احتمال هم وجود دارد که با ورودی های داده شده اصلا نتوانیم آن خروجی را ایجاد کنیم.

۶. چه راه حلی برای تمام شدن برنامه در صورتی که مسئله جواب نداشته باشد پیشنهاد می دهید؟

یکی از راه حل ها می‌تواند استفاده از زمان باشد که اگر از مدت خاصی گذشت به این نتیجه می‌رسیم که مسئله پاسخی با ورودی های داده شده ندارد.

راه حل دیگر این می‌تواند باشد که از ورودی های داده شده استفاده کنیم و حداقل و حداکثر خروجی را بدست بیاوریم بعد اگر در آن بازه نبود متوجه می‌شویم که نمیتوان آن خروجی را ایجاد کرد.

راه دیگری هم وجود دارد. این که با ورودی های داده شده ابتدا تمام خروجی های ممکن را حساب کنیم و ببینیم ایبا جواب خواسته شده وجود دارد یا خیر که بسیار طول خواهد کشید.

گزارش کد :

بخش یک: مشخص کردن مفاهیم اولیه

```
    def __init__(self, operators, operands, equation_length, goal_number):
        self.operators = operators
        self.operands = operands
        self.equation_length = equation_length
        self.goal_number = goal_number
        self.population = self.make_first_population()
```

بخش دو: تولید جمعیت اولیه

```
    def make_first_population(self):
        new_population = []
        for i in range(population_size):
            new_chromosome = ''
            for i in range(self.equation_length):
                new_chromosome += str(self.operands[random.randint(0, len(self.operands) -
1)]) if i%2 == 0 else \
                                  str(self.operators[random.randint(0, len(self.operators)
- 1)])
            new_population.append(new_chromosome)

        return new_population
```

بخش سه: پیاده سازی و مشخص کردن تابع معیار سازگاری

```
    def calc_fitness(self, chromosome):
        return abs(self.goal_number - eval(chromosome))
```

بخش چهار: پیاده سازی mutation و crossover و تولید جمعیت بعدی

```
    def create_crossover_pool(self):
        crossover_pool = []
        candidates = []
        self.create_crossover_chromosomes(crossover_pool, candidates)
```

```
        self.crossover_candidates(candidates)
        crossover_pool.extend(candidates)
        return crossover_pool

    def mutate(self, chromosome):
        chromosome = list(chromosome)
        num_of_mutate = random.randint(1, self.equation_length)
        for i in range(num_of_mutate):
            index = random.randint(1, self.equation_length) - 1
            chromosome[index] = str(random.choice(self.operands)) if
chromosome[index].isnumeric() \
            else str(random.choice(self.operators))
        chromosome = "".join(chromosome)
        return chromosome
```

بخش پنج: ایجاد الگوریتم ژنتیک روی مسئله

```
    def find_equation(self):
        while (True):
            random.shuffle(self.population)
            fitnesses = []
            for i in range(population_size):
                new_eq = self.population[i]
                new_fitt = self.calc_fitness(new_eq)
                fitnesses.append(new_fitt)
                if new_fitt == 0:
                    return new_eq

            best_chromosomes = [x for _,x in sorted(zip(fitnesses, self.population))]
            carried_size = int(population_size*carry_percentage)
            carried_chromosomes = best_chromosomes[:carried_size]

            crossover_pool = self.create_crossover_pool()

            self.population.clear()

            for i in range(population_size - carried_size):
                self.population.append(self.mutate(crossover_pool[i]))

            self.population.extend(carried_chromosomes)
```

گزارش خروجی :

```
C:\Users\ali18\Desktop\5\AI\CAs_and_HWs\CA2>python CA2-810199461.py
21
1 2 3 4 5 6 7 8
+ - *
18019
5*3*5*6*8*5+5-2-8+3*8
```