به نام خدا

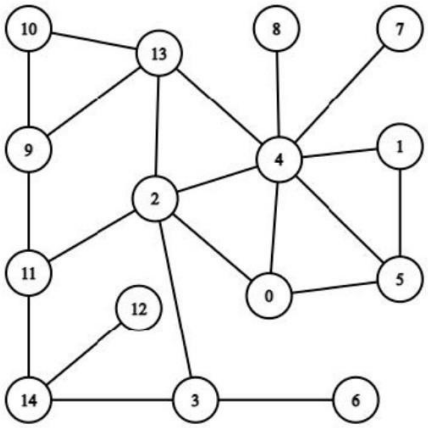
درس هوش مصنوعی و سیستم های خبره

تمرین چهارم

مدرس : دکتر محمدی

دانشجو : سارا سادات یونسی / ۹۸۵۳۳۰۵۳

۱. با توجه به اینکه رنگ آمیزی گراف یک مسئله complete–NP م xباشد، بررس xدرست بودن یک پاسخ برای مسئله آسان م xباشد ول xپیدا کردن پاسخ مورد نظر کاری دشوار است. در این تمرین سع xم xکنیم برای این مسئله یک پاسخ بهینه به کمک الgوریتم ژنتیک پیدا کنیم. (یا حداقل یک bound upper خوبی برای مسئله) گراف زیر را در نظر بgیرید. م xخواهیم با کمترین تعداد رنگ راس های گراف را به گونه ای رنگ کنیم که هیچ دو راس مجاور همرنگ نباشند.



(آ) تولید جمعیت اولیه

در ابتدا با توجه به مسئله و گراف مد نظر جمعیت اولیه را تشکیل دهید. نحوه تولید جمعیت و encoding خود را به طور کامل توضیح دهید.(فرض کنید اندازه جمعیت اولیه برابر با ۶ است.)

جواب (آ)

با توجه به گراف ۱۵ گره داریم که هرکدام رنگ rخواهند داشت بنابراین کروموزوم ۱۵ قسمت خواهد داشت. حال بیشترین تعداد رنگ موجود برای گراف کامل خواهد بود که هر دو گره بهم متصل اند که حداکثر نیاز به ۱۵ رنگ متفاوت خواهد بود. پس میتوان با ۴ بیت تا ۱۶ رنگ برای هر گره کد کرد. به این ترتیب طول کروموزوم برابر با ۶۰ = ۴ * ۱۵ بیت خواهد بود.

برای تولید جمعیت ۶ رشته ۶۰ بیتی به صورت تصادفی تولید می،نیم. از سمت چپ ۴ بیت اول برای گره صفر، ۴ بیت دوم برای گره ۱ و ...

010001010010001010100101001000111000100111111011001000110000
011010100010000110110111010001011000100011101011010001001100
101010101101001001001001001111000000011000101100110000001101
011110110111001000101101101001100101011110001000111011011000
010100101011101010001101010001111001100010111010011100011001
011001100011111001010111110010001101001101110100011101011000

(ب) محاسبهFitness

نحوه محاسبه fitness هر کروموزم (یک عضو جمعیت) را بیان کنید. با محاسبه value fitnessمربوط به هر عضو از جمعیت، اعضا را به ترتیب fit بودن، مرتب کنید. این مقادیر چه چیزی را نشان می دهند؟

جواب (ب)

مقدار fitness را میتوان برای مثال فرمول زیر را برای هر کروموزوم درنظر گرفت:

20 × (number of nodes with correct colors) – (number of distinct colors)

این عدد ترکیبی از تعداد گره های با رنگ درست و تعداد رنگ های متمایز است. به طوری که کروموزومی بیشترین امتیاز را دارد که بیشترین گره های درست با کمترین تنوع رنگی را داشته باشد. لیست مرتب شده رشته ها بصورت زیر است :

291: 10101010110100100100100100111100000000110001011001100000001101

290:01010010101110101000110101000111100110001011101001100011001

289:01101010001000011011011101000101100010001110101101000100110 0

251:01100110001111100101011111001000110100110111010000111010110 00

211:01111011011100100010110110100110010101111000100011101101100 0

190:01000101001000101010010100100011100010011111101100100011000 0

(ج) Mutation و Crossover

این مرحله عملیات تولید نسل جدید را انجام دهید و با ارائه روش های Crossover و mutation روی جمعیت اولیه اجرا کنید و نسل جدید را بدست آورید.

(ج)

برای crossover با احتمال ۶۶ درصدی، ۴ تا از ۶ رشته موجود را به عنوان والد بطور تصادفی انتخاب کرده و برای-crossover point-twoدو نقطه تصادفی در هر رشته انتخاب می کنیم که در رشته های زیر با نقطه مشخص کردیم. ۴ والد بصورت زیر هستند:

291: 1010101011010.0100100100100111100000001100010 1.100110000001101

251: 0110011000111.1100101011111001000110100110111 0.100011101011000

290: 0101001010111010100.0011010100011110011000101110100 11.10001100

190: 0100010100100010100.1001010010001110001001111111011001.000110000

با اعمال crossover فرزندان زیر تولید شدند، همچنین بطور تصادفی در دو رشته دوم و سوم عمل mutation با قرینه کردن بیت اعمال میکنیم

290: 1010101011010.110010101111100100011010011011 10.100110000001101

288:0110011010111.010010010010011111000000010000101.100011101011000

171:0101001010110010 10.100101001000111000100101111011001.100011001

248:0100010100100010 10.001101010001111001100010111010011.000110000

برای انتخاب نسل بعدی برای آنکه ترکیبی از رشته ها با fitness زیاد و کم داشته باشیم ۱۰ رشته نهایی را ۳ تای بالا با fitnessبیشتر را انتخاب میکنیم، همچنین از ۷ رشته بعدی با fitness کمتر ۳ مورد بطور تصادفی انتخاب میکنیم. نسل جدید بصورت زیر است :

291: 10101010110100100100100100111100000001100010110011000000 1101

290:0101001010111010100011010100011110011000101110100111000110 01

290:1010101011010110010101111100100011010011011101001100000011 01

288:0110011010111010010010010011111000000010000101100011101011000

248:0100010100100010100011010100011110011000101110100110001100 00

190:0100010100100010101001010010001110001001111110110010001100 00

(د) نسل جدید

با توجه به آنچه که در قسمت قبل انجام دادید، اکنون باید یک جمعیت جدید داشته باشید. مقادیر Fitness را برای اعضای این جمعیت محاسبه کنید. سپس مجموع Fitnessجمعیت فعل xو رابا مجموع Fitness جمعیت مرحله قبل مقایسه کنید. نتیجه این مقایسه چه چیزی را نشان میدهد؟

(د)

مقدار برای جمعیت قبلی برابر با ۱۵۲۲ و برای جمعیت جدید برابر با ۱۵۹۷ است. یعنی جمعیت رو به بهبود قدم برداشته و به fitnessهای بالاتر نزدیک میشود.

تحلیل نتایج سوال عملی

در این سوال ما باید به پیاده سازی روش hill-climbing در مسئله N-Queen بپردازیم همانگونه که می دانیم الگوریتم hill-climbing تپه‌نوردی برای یافتن بهینه محلی مناسب است اما تضمین نمی‌کند که بهترین راه حل ممکن (بهینه سراسری) از

بین تمام راه‌حل‌های ممکن (فضای جستجو) پیدا شود. در مسائل محدب، تپه‌نوردی بهترین انتخاب است. مثال‌های از الگوریتم‌هایی که مسائل محدب را با تپه‌نوردی حل می‌کنند شامل الگوریتم سیمپلکس برای برنامه‌ریزی خطی، و جستجوی باینری است. اگر محیط جستجو محدب نباشد، این الگوریتم اغلب در یافتن ماکزیموم سراسری شکست خواهد خورد.

گام ۱ : سنجش حالت آغازین ، آیا این حالت، حالت هدف است؟ در این صورت موفقیت حاصل شده و الگوریتم متوقف خواهد شد.

گام ۲ : لوپ تا پیدا شدن پاسخ مورد نظر یا اتمام عملگر ها ادامه خواهد داشت.

گام ۳ : اضافه کردن یک عملگر به حالت کنونی

گام ۴ : چک کردن حالت جدید:

در صورتی که حالت جدید، حالت هدف باشد عملیات متوقف می‌شود.

در صورتی که از حالت فعلی بهتر باشد حالت جدید به حالت فعلی تغییر پیدا خواهد کرد.

در صورتی که حالت جدید از حالت فعلی بهتر نباشد به گام ۲ باز خواهیم گشت.

کد فایل hill_climbing.py

```
import random


def hill_climbing(problem):
    ''' Returns a state as the solution of the problem '''
    current = problem.initial()
    while True:
        neighbours = problem.neighbors(current)
        if not neighbours:
            break
        neighbour = max(neighbours,
                        key=lambda n: (problem.value(n), random.random()))
        if problem.value(neighbour) <= problem.value(current):
            break
        current = neighbour
    return current


def hill_climbing_random_restart(problem, limit=10):
    state = problem.initial()
    cnt = 0
    while problem.goal_test(state) == False and cnt < limit:
        state = hill_climbing(problem)
        cnt += 1
    return state
```

کد فایل N-Queen.py

```
from collections import Counter
from random import randrange


class NQueens:

    def __init__(self, N):
        self.N = N

    def initial(self):
        # Returns a random initial state
        return tuple(randrange(self.N) for i in range(self.N))

    def goal_test(self, state):
        # Returns True if the given state is a goal state
        n = set()
        minus = set()
        add = set()
        for i, j in enumerate(state):
            if j in n:
                return False
            if i - j in minus:
                return False
            if i + j in add:
                return False

            n.add(j)
            minus.add(i - j)
            add.add(i + j)
        return True

    def value(self, state):
        # Returns the value of a state. The higher the value, the closest to
a goal state
        # to calculate number of item are attacking queen
        n, minus, add = Counter(), Counter(), Counter()
        for i, j in enumerate(state):
            n[j] += 1
            minus[i - j] += 1
            add[i + j] += 1
        heuristic = 0
        lst = [n, minus, add]
        for count in lst:
            for i in count:
                heuristic += count[i] * (count[i] - 1) / 2
```

```
        return -heuristic

    def neighbors(self, state):
        # Returns all possible neighbors (next states) of a state
        nearest_list = []
        for i in range(self.N):
            for j in range(self.N):
                if j != state[i]:
                    child = list(state)
                    child[i] = j
                    nearest_list.append(tuple(child))
        return nearest_list
```

کد فایل main.py

```
from time import time

from NQueens import NQueens
from hill_climbing import hill_climbing_random_restart

if __name__ == "__main__":
    print("Running local search for N Queens Problem")
    size = eval(input(" - Please input the size of the board (4~15): "))
    print("\nhill_climbing_random_restart")

    problem = NQueens(size)
    times = 10
    cnt = 0
    start = time()
    for i in range(times):
        result = hill_climbing_random_restart(problem)
        if problem.goal_test(result):
            cnt += 1
    print(" - Accuracy: %2d/%d\tRunning time: %f" % (cnt, times, time() -
start))
```

نتایج حاصل را میبینیم که در مورد ۴ و ۵ که ۱۰/۱۰ شده است :


Terminal: Local (2)
PS D:\Telgram Downloads\HW4_AI> cd .\HW4_AI_Practical\
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 4
hill_climbing_random_restart
- Accuracy: 10/10 Running time: 0.007982
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 5
hill_climbing_random_restart
- Accuracy: 10/10 Running time: 0.015337
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 6
hill_climbing_random_restart
- Accuracy: 7/10 Running time: 0.070036
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 7
hill_climbing_random_restart
- Accuracy: 10/10 Running time: 0.100001
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 8
hill_climbing_random_restart
- Accuracy: 8/10 Running time: 0.249031
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical>
TODO Problems Terminal Python Packages Python Console
Event Log
18:20 LF UTF-8 4 spaces Python 3.9
631 of 2030M


Terminal: Local (2)
- Accuracy: 10/10 Running time: 0.015337
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 6
hill_climbing_random_restart
- Accuracy: 7/10 Running time: 0.070036
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 7
hill_climbing_random_restart
- Accuracy: 10/10 Running time: 0.100001
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 8
hill_climbing_random_restart
- Accuracy: 8/10 Running time: 0.249031
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 9
hill_climbing_random_restart
- Accuracy: 9/10 Running time: 0.332174
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical> python .\main.py
Running local search for N Queens Problem
- Please input the size of the board (4~15): 10
hill_climbing_random_restart
- Accuracy: 7/10 Running time: 0.475157
PS D:\Telgram Downloads\HW4_AI\HW4_AI_Practical>
TODO Problems Terminal Python Packages Python Console
Event Log
18:20 LF UTF-8 4 spaces Python 3.9
716 of 2030M