nVazriGeneticAlgorithm.py > random_pick

You, 2 minutes ago | 2 authors (waqqasiq and others)

```python
import random

def random_chromosome(size): #making random chromosomes
    return [ random.randint(1, n_vazir) for _ in range(n_vazir) ]

def fitness(chromosome):
    horizontal_collisions = sum([chromosome.count(vazir)-1 for vazir in chromosome])/2
    diagonal_collisions = 0

    n = len(chromosome)
    left_diagonal = [0] * 2*n
    right_diagonal = [0] * 2*n
    for i in range(n):
        left_diagonal[i + chromosome[i] - 1] += 1
        right_diagonal[len(chromosome) - i + chromosome[i] - 2] += 1

    diagonal_collisions = 0
    for i in range(2*n-1):
        counter = 0
        if left_diagonal[i] > 1:
            counter += left_diagonal[i]-1
        if right_diagonal[i] > 1:
            counter += right_diagonal[i]-1
        diagonal_collisions += counter / (n-abs(i-n+1))

    return int(maxFitness - (horizontal_collisions + diagonal_collisions)) #28-(2+3)=23

def probability(chromosome, fitness):
    return fitness(chromosome) / maxFitness

def random_pick(population, probabilities):
    populationWithProbabilty = zip(population, probabilities)
    total = sum(w for c, w in populationWithProbabilty)
    r = random.uniform(0, total)
    upto = 0
    for c, w in zip(population, probabilities):
```

waqqasiq, 3 years ago • N-Queen problem/solution coded in python

PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE

TERMINAL

```
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 744 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:3     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:4     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:5     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:6     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:10    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:13    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:15    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:17    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:18    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:19    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:20    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12

Maximum Fitness = 13
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 745 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:3     Chromosome = [2, 2, 5, 1, 3, 1],  Fitness = 12
i:4     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:5     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:6     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:10    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:13    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:15    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
```

OUTPUT

nVazriGeneticAlgorithm.py > random_pick

```
    r = random.uniform(0, total)
    upto = 0      waqqasiq, 3 years ago • N-Queen problem/solution coded in python
    for c, w in zip(population, probabilities):
        if upto + w >= r:
            return c
        upto += w
    assert False, "Shouldn't get here"

def reproduce(x, y): #doing cross_over between two chromosomes
    n = len(x)
    c = random.randint(0, n - 1)
    return x[0:c] + y[c:n]

def mutate(x):  #randomly changing the value of a random index of a chromosome
    n = len(x)
    c = random.randint(0, n - 1)
    m = random.randint(1, n)
    x[c] = m
    return x

def genetic_vazir(population, fitness, m_p):
    mutation_probability = m_p
    new_population = []
    probabilities = [probability(n, fitness) for n in population]
    for i in range(len(population)):
        x = random_pick(population, probabilities) #best chromosome 1
        y = random_pick(population, probabilities) #best chromosome 2
        child = reproduce(x, y) #creating two new chromosomes from the best 2 chromosomes
        if random.random() < mutation_probability:
            child = mutate(child)
        print('i:{}'.format(i+1), end="\t")
        print_chromosome(child)
        new_population.append(child)
        if fitness(child) == maxFitness: break
    return new_population

def print_chromosome(chrom):
```

PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE

TERMINAL

```
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 744 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:3     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:4     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:5     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:6     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:10    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:13    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:15    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:17    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:18    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:19    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:20    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12

Maximum Fitness = 13
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 745 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:3     Chromosome = [2, 2, 5, 1, 3, 1],  Fitness = 12
i:4     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:5     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:6     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:10    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:13    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:15    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
```

OUTPUT

nVazriGeneticAlgorithm.py > random_pick

```python
def print_chromosome(chrom):
    print("Chromosome = {},  Fitness = {}"
        .format(str(chrom), fitness(chrom)))

if __name__ == "__main__":
    n_vazir = int(input("Tedad Vazir: "))                  # N V = 6
    n_population = int(input("Jamiat har generation: "))   # N Population = 20
    mutation_probability = float(input("Mutation Probability (n<1 ==> 0.03): "))    # Mutation Probabil
    maxFitness = (n_vazir*(n_vazir-1))/2                   # maxFitness = 6*5/2 = 15
    population = [random_chromosome(n_vazir) for _ in range(n_population)]

    generation = 1
    while not maxFitness in [fitness(chrom) for chrom in population]:
        print("|=== Generation {} ===|".format(generation))
        population = genetic_vazir(population, fitness, mutation_probability)
        print("")
        print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))
        generation += 1
        print("|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|")
    chrom_out = []
    print("Solved in Generation {}!".format(generation-1))
    for chrom in population:
        if fitness(chrom) == maxFitness:
            print("");
            print("One of the solutions: ")
            chrom_out = chrom
            print_chromosome(chrom)

    board = []

    for x in range(n_vazir):
        board.append(["x"] * n_vazir)

    for i in range(n_vazir):
        board[n_vazir-chrom_out[i]][i]="V"
```

PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE

TERMINAL

```
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 744 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:3     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:4     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:5     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:6     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:10    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:13    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:15    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:17    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:18    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:19    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:20    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12

Maximum Fitness = 13
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 745 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:3     Chromosome = [2, 2, 5, 1, 3, 1],  Fitness = 12
i:4     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:5     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:6     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:10    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:13    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:15    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
```

OUTPUT

nVazriGeneticAlgorithm.py > random_pick

```
    generation = 1
    while not maxFitness in [fitness(chrom) for chrom in population]:
        print("|=== Generation {} ===|".format(generation))
        population = genetic_vazir(population, fitness, mutation_probability)
        print("")
        print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))
        generation += 1
        print("|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|")
    chrom_out = []
    print("Solved in Generation {}!".format(generation-1))
    for chrom in population:
        if fitness(chrom) == maxFitness:
            print("");
            print("One of the solutions: ")
            chrom_out = chrom
            print_chromosome(chrom)

    board = []

    for x in range(n_vazir):
        board.append(["x"] * n_vazir)

    for i in range(n_vazir):
        board[n_vazir-chrom_out[i]][i]="V"


    def print_board(board):
        for row in board:
            print (" ".join(row))

    print()
    print_board(board)
    print()
    print("Forked from {https://github.com/waqqasiq/n-queen-problem-using-genetic-algorithm}")
```

PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE

TERMINAL PYTHON

```
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 744 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:3     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:4     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:5     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:6     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:10    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:13    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:15    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:17    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:18    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:19    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:20    Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12

Maximum Fitness = 13
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 745 ===|
i:1     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:2     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:3     Chromosome = [2, 2, 5, 1, 3, 1],  Fitness = 12
i:4     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:5     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:6     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:7     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:8     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:9     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:10    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:11    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:12    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:13    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:15    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
```

OUTPUT

nVazriGeneticAlgorithm.py > random_pick

```
    generation = 1
    while not maxFitness in [fitness(chrom) for chrom in population]:
        print("|=== Generation {} ===|".format(generation))
        population = genetic_vazir(population, fitness, mutation_probability)
        print("")
        print("Maximum Fitness = {}".format(max([fitness(n) for n in population])))
        generation += 1
        print("|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|")
    chrom_out = []
    print("Solved in Generation {}!".format(generation-1))
    for chrom in population:
        if fitness(chrom) == maxFitness:
            print("");
            print("One of the solutions: ")
            chrom_out = chrom
            print_chromosome(chrom)

    board = []

    for x in range(n_vazir):
        board.append(["x"] * n_vazir)

    for i in range(n_vazir):
        board[n_vazir-chrom_out[i]][i]="V"


    def print_board(board):
        for row in board:
            print (" ".join(row))

    print()
    print_board(board)
    print()
    print("Forked from {https://github.com/waqqasiq/n-queen-problem-using-genetic-algorithm}")
```

PROBLEMS OUTPUT DEBUG CONSOLE SQL CONSOLE

TERMINAL

```
i:12    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:13    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:14    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:15    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:16    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:17    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:18    Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:19    Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:20    Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13

Maximum Fitness = 13
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
|=== Generation 746 ===|
i:1     Chromosome = [2, 2, 5, 1, 3, 5],  Fitness = 12
i:2     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:3     Chromosome = [2, 4, 5, 1, 3, 5],  Fitness = 13
i:4     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:5     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:6     Chromosome = [2, 4, 5, 1, 3, 1],  Fitness = 13
i:7     Chromosome = [2, 4, 5, 1, 1, 5],  Fitness = 12
i:8     Chromosome = [2, 4, 5, 1, 1, 1],  Fitness = 11
i:9     Chromosome = [2, 4, 6, 1, 3, 5],  Fitness = 15

Maximum Fitness = 15
|=====|=====|=====|=====|=====|=====|=====|=====|=====|=====|
Solved in Generation 746!

One of the solutions:
Chromosome = [2, 4, 6, 1, 3, 5],  Fitness = 15

x x V x x x
x x x x x V
x V x x x x
x x x x V x
V x x x x x
x x x V x x

Forked from {https://github.com/waqqasiq/n-queen-problem-using-genetic-algorithm}
zakaria@legionrig:~/Documents/University/Semister 04/Artificial-Intelligence/HomeWork/nVazir$
```

OUTPUT