presentation

December 16, 2020

1 *Aplicação de PSO :*

2 *Caixeiro Viajante*

2.1 *Overview*

2.1.1 PSO Discreto

- *Posição da Partícula:* Uma rota válida
- *Velocidade:* N par de trocas simples entre elementos da rota

2.1.2 Partícula

[ ]:
```
import numpy as np
class DParticle:
    def __init__(self, path: np.array):
        self.position = path
        self.combination_count = np.random.randint(len(path) * (len(path) - 1))␣
↪+ 1
        self.velocity = np.random.randint(len(path), size=(self.
↪combination_count, 2))
        self.best_position = np.copy(self.position)
        self.best_path_len = np.inf

    def __repr__(self):
        return self.__str__()
```

2.1.3 Fit

[ ]:
```
def fit(path, problem):
    cyclic_path = np.hstack((path, np.array([path[0]])))
    return sum(problem.get_weight(a, b) for a, b in zip(cyclic_path[0:],␣
↪cyclic_path[1:]))
```

2.1.4 Atualização

[ ]:
```
def discrete_velocity(particle: DParticle):
    return random.choices(particle.velocity, k=np.random.randint(len(particle.
 ↪position)))
```

2.1.5 Algoritmo PSO Discreto

[ ]:
```
def submit(self, iterations=1000):
    for i in range(iterations):
        for particle in self.particles:
            velocity = discrete_velocity(particle)
            adjust_discrete_position(particle, velocity)
            distance = fit(particle.position, self.problem)

            logger.debug(f"Distance: {distance}\t Path:{particle.position}\tV:
 ↪{velocity}")

            # Is it the best particle distance so far?
            if distance < particle.best_path_len:
                particle.best_position = np.copy(particle.position)
                particle.best_path_len = distance
                # May be the best global distance as well?
                if distance < self.best_path:
                    self.best_path = distance
                    self.best_path_pos = np.copy(particle.position)
                    logger.info(f"B.D: {self.best_path}\tB.P:{self.
 ↪best_path_pos}")
```

[ ]:
```
def adjust_discrete_position(particle: DParticle, velocity: np.array):
    for exchange in velocity:
        tmp = np.copy(particle.position[exchange[0]])
        particle.position[exchange[0]] = particle.position[exchange[1]]
        particle.position[exchange[1]] = tmp
```

2.2 Resultados

[6]:
```
import pandas as pd
df = pd.read_csv("benchmark.csv")
```

	algoritmo	problem	mean	min
0	Ideal	24	-	1272
1	AG Sugerido	24	1331	1272
2	AG Desenvolvido	24	-	1300
3	PSO Continuo	24	2148.3	2021
4	PSO Discreto	24	2125.6	1813

	algoritmo	problem	mean	min
5	Ideal	48	-	5046
6	AG Sugerido	48	5533	5080
7	AG Desenvolvido	48	-	6893
8	PSO Continuo	48	15801	14630
9	PSO Discreto	48	16074	15422

2.3 Referências

[1] Codigo em Python de um exemplo com PSO, http://paginapessoal.utfpr.edu.br/cesarbenitez/algoritmos-evolutivos/Exemplo_PSO_Rastrigin_python.txt.txt/view

[2] Inteligência de Enxame: PSO, http://www.eng.uerj.br/~nadia/pso.pdf