Resolvendo o problema do Caixeiro Viajante utilizando o algoritmo de enxame de abelhas ABC

Sep 10, 2018

Algoritmo de Enxame de abelhas ideia princípal

Inspirado pelo comportamento inteligente de forrageamento (i.e., busca por alimento) dos enxames de abelhas produtoras de mel, o algoritmo de colônia de abelhas (ABC)(do inglês, Artificial Bee Colony Algorithm) é uma meta-heurística baseada em população (KARABOGA et al. ,2014). O ABC foi proposto inicialmente por Karaboga (2005) para solucionar problemas de otimização contínua e posteriormente também foi adaptado para problemas de domínio discreto (

KARABOGA , 2005; KARABOGA; BASTURK , 2007, 2008; KARABOGA ,2009; KARABOGA; AKAY , 2009; KARABOGA et al. , 2014).

Karaboga (2005) descreve em seu trabalho que o algoritmo ABC é composto por três tipos de abelhas, sendo elas: empregadas, espectadoras e a exploradora. A função da abelha empregada é a de explorar fontes de alimentos em sua vizinhança (i.e., busca local). A abelha que fica na colmeia esperando para tomar a decisão de qual fonte de alimento escolher para explorar é conhecida como espectadora (i.e., busca local). A abelha que sai a procura de novas fontes de alimentos de forma aleatória é a exploradora (i.e., busca global).

A colmeia de abelhas do algoritmo ABC é composta por 50% de abelhas empregadas e espectadoras e 1 abelha exploradora. No processo de forrageamento do ABC inicialmente as abelhas empregadas são enviadas para as fontes de alimento para a coleta de néctar. Depois de recolher o alimento ela volta para a colmeia e compartilha as informações da localização dessa fonte de alimento para a abelha espectadora. Esse processo de comunicação entre essas abelhas é conhecido como dança do requebrado (do inglês, wiggle dance). Então, após uma fonte de alimento se esvaziar a abelha espectadora que era designada aquela fonte de alimento se torna uma abelha exploradora e sai para fora da colmeia em busca de novas fontes de alimento.

Com base no contexto supracitado, Karaboga (2005) desenvolveu a ideia do algoritmo ABC. Para mais detalhes eu sugiro que você faça a leitura do artigo do profº Karaboga de 2005.

Antes de iniciar os processos a serem apresentados a seguir, se você desejar, seria interessante o uso de um ambiente virtual fechado. Para isso você pode consultar esse link para o artigo para ver como utilizar um ambiente virtual fechado em projetos python. Se não, pode continuar a leitura.

Aplicando os conceitos do algoritmo ABC no problema do CV

O problema do Caixeiro Viajante (CV) representa a ideia de um CV que tem de visitar um conjunto de cidades sem repetir nenhuma delas de modo que ele retorne a cidade inicial por meio da menor distância possível.

O primeiro passo para trabalhar com o problema do CV é realizar a leitura das coordenadas das cidades e gerar a tabela de distâncias referente a essas cidades.

Antes de começar o passo a passo, é importante salientar que será utilizado no decorrer do artigo a linguagem de programação **python** na versão **3** e será necessário instalar algumas bibliotecas sendo elas a **numpy** e a **matplotlib**. Caso você não as tenha instaladas no seu

computador a seguir é ilustrado um exemplo dos comandos necessários para instalar as bibliotecas mencionadas.

```
    Abra seu terminal e digite os comandos:
    pip install numpy
    pip install matplotlib
```

Depois de instalar as bibliotecas utilizadas faça o seguinte:

- Abra seu editor de texto favorito e crie o seguinte arquivo rafael5.txt;
- Em seguida salve o seguinte conteudo dentro do arquivo criado:

```
1 37 52
2 49 49
3 52 64
4 20 26
5 40 30
```

- No bloco acima, a primeira coluna representa **a cidade**, a segunda coluna **a coordenada x da cidade** e a terceira coluna **a coordenada y da cidade**;
- No meu caso, eu criei um diretório (i.e., uma pasta) chamado “tsp-instancias” e dentro dele eu coloquei o arquivo rafael5.txt;
- No mesmo nível do diretório “tsp-instancias” crie um arquivo python chamado abc.py.

```
├── abc.py
└── tsp-instancias
    └── rafael5.txt
```

No arquivo abc.py você irá inserir os códigos apresentados no decorrer do artigo para implementar o algoritmo ABC.

Para realizar a leitura dos dados por meio de um arquivo externo (i.e., nosso arquivo rafael5.txt que tem os dados do problema do cv), utiliza-se a seguinte forma a baixo:

```
1
2
3
4
5
6
7
```

```
    import numpy as np #biblioteca que facilita o armazenamento
    import math # fornece funcões matemáticas
    import matplotlib.pyplot as plt #facilita e prove a geração

    instancia     = 'tsp-instancias/rafael5.txt'
    cidades       = np.genfromtxt(instancia, delimiter=' ', use
```

```
    coordenadas_x = np.genfromtxt(instancia, delimiter=' ', use
    coordenadas_y = np.genfromtxt(instancia, delimiter=' ', use

    numero_cidades = len(cidades)

    fig, ax = plt.subplots()

    for i, txt in enumerate(cidades):
        ax.annotate(int(txt), (coordenadas_x[i], coordenadas_y
        plt.xlabel('Coordenadas X')
    plt.ylabel('Coordenadas Y')
    plt.title('Coordenadas das cidades utilizadas no CV')

    plt.plot(coordenadas_x,coordenadas_y,'ro')
    plt.show()
```

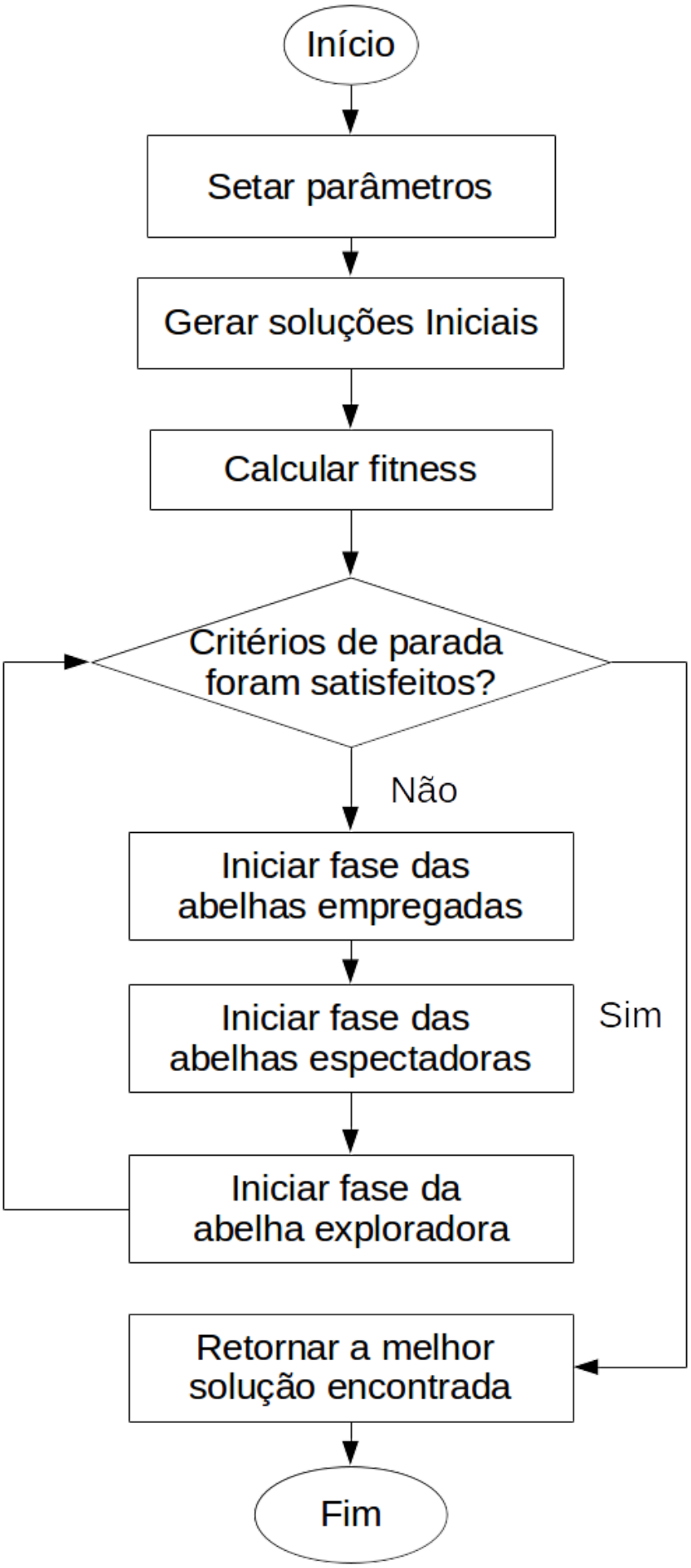
Para gerar a tabela de que representa a distância gasta entre as cidades utiliza-se a função a seguir:

```
    #com os dados das coordenadas das cidades, aplicar a formul
    def distancia_euclidiana(x,y):
        distancias = np.zeros([numero_cidades,numero_cidades])
        for i in range(len(x)):
            for j in range(len(y)):
                #aplicando a distancia eucliana nas cidades i e
                distancias[i][j] = math.sqrt( math.pow( (x[i]-x
        return distancias
    #aqui ficaram armazenadas as distâncias que o cv vai gastar
    distancias = distancia_euclidiana(coordenadas_x,coordenadas

print(distancias) #imprimindo na tela a matriz gerada com as di
```

Com os dados básicos do problema do CV, agora já é possível começar a pensar no algoritmo ABC de fato para realizar a otimização do problema.

Fluxograma do algoritmo ABC


Início
Setar parâmetros
Gerar soluções Iniciais
Calcular fitness
Critérios de parada foram satisfeitos?
Não
Sim
Iniciar fase das abelhas empregadas
Iniciar fase das abelhas espectadoras
Iniciar fase da abelha exploradora
Retornar a melhor solução encontrada
Fim

Modelagem da Solução (i.e., representação das variáveis de decisão do problema estudado)

A modelagem da solução é a forma com a qual seu algoritmo ABC representa as variáveis de decisão do problema de otimização em questão. No caso do CV a representação utilizada será a seguinte:

Cada abelha terá como solução uma sequência de cidades e.g., (1, 2, 3, …, até N) sendo que N representa a quantidade de cidades do nosso problema/exemplo e nesse caso, N=5 de modo que elas não se repitam. O fato delas não se repetirem é considerado como uma restrição do problema em questão.

SETAR OS PARÂMETROS DO ABC

```
### SETAR PARÂMETROS INICIO ###
#Configuração/Parametrização do algoritmo ABC
tamanhoDaColonia = 10 # tamanho total da colônia de abelhas (em
metadeDaColonia = tamanhoDaColonia/2 #referente ao tamanho da m
numeroDeTentativas = 10 # número de tantivas que relacionadas a
D = numero_cidades #dimensionalidade do problema em questão ou
numeroDeCiclosDeForrageamento = 50 #número de iterações realiza
numeroDeExecucoes = 10 #número de execuções realizadas pelo alg

colonia = np.zeros([metadeDaColonia,D], dtype=int) #criação da
tentativas = np.zeros([metadeDaColonia]) #array que armazenara
fitnessDaColonia = np.zeros([metadeDaColonia]) #array que armaz


melhorSolucao = np.zeros([D], dtype=int) #melhor solução atual
melhorFitness = 0 #valor de fitness da melhor solução atual

melhoresSolucoes = np.zeros([numeroDeExecucoes,D], dtype=int) #
melhoresFitness  = np.zeros([numeroDeExecucoes]) #array referen
### SETAR PARÂMETROS FIM ###
```

FUNÇÃO DE APTIDÃO/FITNESS UTILIZADA PARA AVALIAR AS SOLUÇÕES DO ABC NO PROBLEMA DO CV

No problema do CV a função de fitness avalia a sequência das cidades representada por cada solução gerenciadas por cada abelha.

```
# função de fitness utilizada para mensurar a qualidade das solu
def fitness(solucao,distancias):
    valorDoFitness = 0
    for i in range(len(solucao)-1):
        valorDoFitness += distancias[solucao[i],solucao[i+1]]
    valorDoFitness += distancias[solucao[-1],solucao[0]] #retorn
    return valorDoFitness
```

CRIAÇÃO DA POPULAÇÃO INICIAL DE ABELHAS DO ENXAME NO ABC PARA O PROBLEMA DO CV

Assim que as soluções iniciais são geradas elas são avaliadas para que se possa saber a qualidade da fonte de alimentos de cada abelha.

```
#gerar população inicial de abelhas
for i in range(metadeDaColonia):
    colonia[i,:] = np.random.permutation(D) #para cada abelha ge
    fitnessDaColonia[i] = fitness(colonia[i],distancias) #calcul
```

MÉTODO DE SELEÇÃO UTILIZADO NO ABC

Depois que todas as abelhas empregadas trabalham nas suas respectivas fontes de alimentos, um método de seleção é utilizado para selecionar uma abelha espectadora para melhorar cada

solução de cada abelha empregada. Para isso, pode-se utilizar o método de Roleta Probabilística. Não existe apenas esse método e se você quiser pode pesquisar por outros e utilizá-los em seu lugar para ver o que acontece com o resultado gerado pelo ABC.

```
def selecaoPorRoleta(fitness): #Roleta probabilista refente as a
    probs = fitness[:]/fitness.sum()
    escolhida = np.random.uniform(0,probs.sum(),1)
    for i in range(len(probs)):
        if probs[i] > escolhida:
            return i
    return -1
```

A TROCA SWAP É APENAS UM MÉTODO QUE ALTERA AS SOLUÇÕES/FONTES DE ALIMENTOS DAS ABELHAS DO ABC

```
def trocaSwap(novaSolucao):
    posicaoDeTrocas = np.random.randint(D, size=(2)) #gerando 2
    while posicaoDeTrocas[0] == posicaoDeTrocas[1]: #garantindo
        posicaoDeTrocas = np.random.randint(D, size=(2))
    aux = novaSolucao[posicaoDeTrocas[0]]
    novaSolucao[posicaoDeTrocas[0]] = novaSolucao[posicaoDeTroca
    novaSolucao[posicaoDeTrocas[1]] = aux
    return novaSolucao
```

A seguir é apresentado o fluxo do ABC do mesmo modo que no fluxograma supracitado nesse artigo:

```
#definir quem é a abelha com melhor fonte de alimento
melhorSolucao[:] = colonia[fitnessDaColonia.argmin()]
melhorFitness = fitnessDaColonia[fitnessDaColonia.argmin()]
```

```
4
5       for r in range(numeroDeExecucoes):
6           #print("Execucao numero: %d" % (r))
7           for iteracao in range(numeroDeCiclosDeForrageamento): #crit
8               #print("Iteracao: %d" % (iteracao))
9               #iniciar fase das abelhas empregadas
10              for i in range(metadeDaColonia):
11                  #gerar uma perturbação na solucao da abelha i e ver
12                  novaSolucao = np.zeros([D], dtype=int)
13                  #aplicando o método simples de troca swap
14                  novaSolucao[:] = trocaSwap(colonia[i])
15                  #verificar se a nova solucao tem melhor aptidão que
16                  if fitness(novaSolucao,distancias) < fitnessDaColon
17                      colonia[i,:] = novaSolucao
18                      fitnessDaColonia[i] = fitness(novaSolucao,dista
19                      tentativas[i] = 0 # se a solução foi melhorada
20                  else:
21                      tentativas[i] += 1 # se a solução da abelha i n
22
23              #iniciar fase das abelhas espectadoras
24              for j in range(metadeDaColonia):
25                  abelhaEmpregadaEscolhida = selecaoPorRoleta(fitness
26                  while abelhaEmpregadaEscolhida == -1:
27                      abelhaEmpregadaEscolhida = selecaoPorRoleta(fit
28                  #gerar uma perturbação na solucao da abelha i e ver
29                  novaSolucao = np.zeros([D], dtype=int)
30                  #aplicando o método simples de troca swap na abelha
31                  novaSolucao[:] = trocaSwap(colonia[abelhaEmpregadaE
32                  #verificar se a nova solucao tem melhor aptidão que
33                  if fitness(novaSolucao,distancias) < fitnessDaColon
34                      colonia[i,:] = novaSolucao
35                      fitnessDaColonia[i] = fitness(novaSolucao,dista
36                      tentativas[i] = 0 # se a solução foi melhorada
37                  else:
38                      tentativas[i] += 1 # se a solução da abelha i n
39
40              #iniciar fase da abelha exploradora - só existe uma abe
41              #primeiro deve se salvar a melhor abelha do ciclo de fo
42              if fitnessDaColonia[fitnessDaColonia.argmin()] < melhor
43                  melhorSolucao[:] = colonia[fitnessDaColonia.argmin(
44                  melhorFitness = fitnessDaColonia[fitnessDaColonia.a
45              #agora que a melhor solução da iteração foi salva verif
46              if tentativas[tentativas.argmax()] >= numeroDeTentativa
47                  #a função da abelha exploradora é encontrar um nova
```

```
48                colonia[tentativas.argmax()] = np.random.permutatio
49                fitnessDaColonia[tentativas.argmax()] = fitness(col
50                tentativas[tentativas.argmax()] = 0 #reseta o núme
51
52            #print("Melhor Solucao e Melhor Fitness Atual")
53            #print(melhorSolucao)
54            #print(melhorFitness)
55
56        melhoresSolucoes[r,:] = melhorSolucao
57        melhoresFitness[r] = melhorFitness
58
59    print("Fim da Execução!")
60    for i in range(numeroDeExecucoes):
61        print("execucao %d" % (i))
62        print(melhoresSolucoes[i,:])
63        print(melhoresFitness[i])
64        print("-----------------------------")
```

Essa parte final serve para plotar em um gráfico o resultado gerado por meio do ABC:

```
1   #Plotando o resultado gerado pelo ABC
2   fig2, ax2 = plt.subplots()
3
4   for i, txt in enumerate(cidades):
5       ax2.annotate(int(txt), (coordenadas_x[i], coordenadas_y[i])
6
7       plt.xlabel('Coordenadas X')
8   plt.ylabel('Coordenadas Y')
9   plt.title('Instância com 5 cidades')
10
11  melhorSolucao = np.append(melhorSolucao[:],melhorSolucao[0])
12  plt.plot(coordenadas_x[melhorSolucao[:]],coordenadas_y[melhorSo
13  plt.show()
14
15  print("Roteiro de cidades percorridas pelo CV = %s" % str(melho
16  print("Distancia percorrida pelo CV = %f" % melhorFitness)
```

Se tiver alguma dúvida pode deixar ela nos comentários que responderei assim que puder!

Se achou algum erro no artigo ou que faltou falar de algo ou gostaria de ver algum tipo de assunto publicado no blog que seja sobre computação em geral ou algo específico sobre inteligência artificial e assuntos relacionados fique a vontade para deixar uma mensagem e também comentar sobre esse artigo. Colabore, CLIQUE AQUI!

Compartilhe esse artigo nas mídias sociais → **Twitter**, **Facebook**, **Google+**,

Comentários

Logar

Não há comentários postados até o momento. Seja o primeiro!

Postar um novo comentário

Digite o texto aqui!

Comentar como Visitante, ou logar: facebook

Nome | Email | Website (opcional)

Mostrar junto aos seus comentários. *Não mostrado publicamente.* *Se você tem um website, linke para ele aqui.*

Assinar Nada

Enviar Comentário

< Tudo Programado />

< Tudo Programado />

rafaelsanches123@gmail.com

rafael.sanches.566

rafaelsanches123

rafaelsanches

Esse blog foi desenvolvido com o intuito de que eu pudesse compartilhar um pouco do meu conhecimento sobre a área de informática com todos. Eu irei compartilhar informação sobre diversos tópicos da computação mas, com foco em especial em

rafaelsanches__
rafael-sanches-24a2b034
rafaelsanches56
RafaelFVSanches
rafaesanches

assuntos referentes a Inteligência Artificial.