



Join GitHub today

Dismiss

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Branch: master ▾

03MAIR---Algoritmos-de-Optimizacion---2019 / AG3 /

Find file

Copy path

WilliamSteveRodriguezVillamizar_AG3.ipynb



wisrovi Creado con Colaboratory

f38cdae 2 minutes ago

1 contributor

629 lines (629 sloc) | 24.6 KB



Raw

Blame

History



AG3- Actividad Guiada 3

Nombre y Apellidos: William Steve Rodriguez Villamizar

Colab: https://colab.research.google.com/github/wisrovi/03MAIR---Algoritmos-de-Optimizacion--2019/blob/master/AG3/WilliamSteveRodriguezVillamizar_AG3.ipynb

Url: https://github.com/wisrovi/03MAIR---Algoritmos-de-Optimizacion--2019/blob/master/AG3/WilliamSteveRodriguezVillamizar_AG3.ipynb

```
In [1]: !pip install request  
!pip install imgaug  
!pip install tsplib95
```

```
Collecting request  
  Downloading https://files.pythonhosted.org/packages/f1/27/7cbde262d854aedef217061a97020d66a63163c5c04e0ec02ff98c5d8f44e/request-2019.4.13.tar.gz  
Collecting get (from request)  
  Downloading https://files.pythonhosted.org/packages/3f/ef/bb46f77f7220ac1b7edba0c76d810c89fddb24ddd8c08f337b9b4a618db7/get-2019.4.13.tar.gz  
Collecting post (from request)  
  Downloading https://files.pythonhosted.org/packages/0f/05/bd79da5849ea6a92485ed7029ef97b1b75e55c26bc0ed3a7ec769af666f3/post-2019.4.13.tar.gz  
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from request) (41.2.0)  
Collecting query_string (from get->request)  
  Downloading https://files.pythonhosted.org/packages/12/3c/412a45daf5bea9b1d06d7de41787ec4168001dfa418db7ec8723356b119f/query-string-2019.4.13.tar.gz  
Collecting public (from query_string->get->request)  
  Downloading https://files.pythonhosted.org/packages/54/4d/b40004cc6c07665e48af22cfe1e631f219bf4282e15fa76a5b6364f6885c/public-2019.4.13.tar.gz  
Building wheels for collected packages: request, get, post, query-string, public  
  Building wheel for request (setup.py) ... done  
  Created wheel for request: filename=request-2019.4.13-cp36-none-any.whl size=1676 sha256=e159f889dd950a25ba0c9fdc356130ad017bb4c7f2ead513c231ffb21a755244
```

```

.....
Stored in directory: /root/.cache/pip/wheels/30/84/5f/484cfba678967ef58c16fce6890925d5c7
172622f20111fbfd
Building wheel for get (setup.py) ... done
Created wheel for get: filename=get-2019.4.13-cp36-none-any.whl size=1692 sha256=ccfaeaa
6a55b595245c011898fe51e3469ff5a6fa8543efa6a19ba64c009671b
Stored in directory: /root/.cache/pip/wheels/c1/e3/c1/d02c8c58538853e4c9b78cadb74f6d5c5c
370b48a69a7271aa
Building wheel for post (setup.py) ... done
Created wheel for post: filename=post-2019.4.13-cp36-none-any.whl size=1661 sha256=28798
df1207866be8b4c59a29214af00ed7ddcf6ddb462f4cb9c3f4d7b9cf703
Stored in directory: /root/.cache/pip/wheels/c3/c3/24/b5c132b537ab380c02d69e6bd4dec1f5db
56b5fe19030473d7
Building wheel for query-string (setup.py) ... done
Created wheel for query-string: filename=query_string-2019.4.13-cp36-none-any.whl size=2
049 sha256=f0d792631eab28fd597b25162c4d7c1ca797988ccb6f86475b5339a0ad7945e5
Stored in directory: /root/.cache/pip/wheels/d6/a4/78/01b20a9dc224dcc009fab669f7f27b943b
8889c5150bd68d8a
Building wheel for public (setup.py) ... done
Created wheel for public: filename=public-2019.4.13-cp36-none-any.whl size=2536 sha256=3
eb910460316elc79adee8264794456eedc5cd3669f4501c10648eb9c036ac86
Stored in directory: /root/.cache/pip/wheels/23/7c/6e/f5b4e09d6596c8b8802b347e48f149031e
2363368048f1347a
Successfully built request get post query-string public
Installing collected packages: public, query-string, get, post, request
Successfully installed get-2019.4.13 post-2019.4.13 public-2019.4.13 query-string-2019.4.1
3 request-2019.4.13
Requirement already satisfied: imgaug in /usr/local/lib/python3.6/dist-packages (0.2.9)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.6/dist-packages (fr
om imgaug) (1.16.5)
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from imga
ug) (4.3.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from
imgaug) (3.0.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from imgaug)
(1.12.0)
Requirement already satisfied: Shapely in /usr/local/lib/python3.6/dist-packages (from img
aug) (1.6.4.post2)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from imgau
g) (1.3.1)
Requirement already satisfied: imageio in /usr/local/lib/python3.6/dist-packages (from img

```

```

Requirement already satisfied: imageio in /usr/local/lib/python3.6/dist-packages (from imgaug) (2.4.1)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-packages (from imgaug) (3.4.5.20)
Requirement already satisfied: scikit-image>=0.11.0 in /usr/local/lib/python3.6/dist-packages (from imgaug) (0.15.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-packages (from Pillow->imgaug) (0.46)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->imgaug) (2.5.3)
Requirement already satisfied: cycycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib->imgaug) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->imgaug) (2.4.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib->imgaug) (1.1.0)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.11.0->imgaug) (2.3)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.11.0->imgaug) (1.0.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotlib->imgaug) (41.2.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx>=2.0->scikit-image>=0.11.0->imgaug) (4.4.0)
Collecting tsplib95
  Downloading https://files.pythonhosted.org/packages/90/9f/5fbf6118d00719cc4688b175a04da09b89c3780db6b0c55bc646a20a6a07/tsplib95-0.3.3-py2.py3-none-any.whl
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.6/dist-packages (from tsplib95) (7.0)
Collecting networkx==2.1 (from tsplib95)
  Downloading https://files.pythonhosted.org/packages/11/42/f951cc6838a4dff6ce57211c4d7f8444809ccbe2134179950301e5c4c83c/networkx-2.1.zip (1.6MB)
    |████████████████████| 1.6MB 4.1MB/s
Requirement already satisfied: decorator>=4.1.0 in /usr/local/lib/python3.6/dist-packages (from networkx==2.1->tsplib95) (4.4.0)
Building wheels for collected packages: networkx
  Building wheel for networkx (setup.py) ... done
  Created wheel for networkx: filename=networkx-2.1-py2.py3-none-any.whl size=1447766 sha256=632a33349f7e15cd045cdbf53494ddaad40e07f0d3904496615958ca09fd8185
  Stored in directory: /root/.cache/pip/wheels/44/c0/34/6f98693a554301bdb405f8d65d95bbcd3e50180c7fdd08a04e

```

```
Successfully built networkx
ERROR: alumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have imgaug
0.2.9 which is incompatible.
Installing collected packages: networkx, tsplib95
  Found existing installation: networkx 2.3
    Uninstalling networkx-2.3:
      Successfully uninstalled networkx-2.3
Successfully installed networkx-2.1 tsplib95-0.3.3
```

```
In [2]: import urllib.request
file = "swiss42.tsp"

urllib.request.urlretrieve("http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/swiss42.tsp"
, file)
```

```
Out[2]: ('swiss42.tsp', <http.client.HTTPMessage at 0x7f18d3c6b7b8>)
```

```
In [0]: import tsplib95
import random
from math import e
```

```
In [4]: problem = tsplib95.load_problem(file)
Nodos = list(problem.get_nodes())
aristas = list(problem.get_edges())

problem.wfunc(0,1)
```

```
Out[4]: 15
```

```
In [0]: def distancia(a,b, problem):
        return problem.wfunc(a,b)

def crear_solucion(Nodos):
    solucion = [0]
    for _ in range(len(Nodos)-1):
        solucion = solucion + [random.choice(list(set(Nodos) - set({0}) - set(solucion)))]
    return solucion

def distancia total(solucion, problem):
```

```

def distancia_total(solucion, problem):
    distancia_total = 0
    for i in range(len(solucion)-1):
        distancia_total += distancia(solucion[i], solucion[i+1], problem)
    return distancia_total + distancia(solucion[len(solucion)-1], solucion[0], problem)

```

In [6]: `solucion = crear_solucion(Nodos)`

```

print("Solución: ", end="")
print(solucion)

distancia_recorrida = distancia_total(solucion, problem)
print("Distancia recorrida: ", end="")
print(distancia_recorrida)

```

Solución: [0, 6, 29, 32, 11, 13, 5, 16, 17, 36, 20, 37, 22, 39, 3, 28, 27, 7, 34, 4, 10, 3, 30, 8, 31, 14, 1, 38, 19, 35, 9, 12, 40, 18, 23, 21, 15, 25, 26, 2, 24, 41]
 Distancia recorrida: 4427

In [0]: `def busqueda_aleatoria(problem, N):`
`Nodos = list(problem.get_nodes())`

```

mejor_solucion = []
mejor_distancia = 10e100

for i in range(N):
    solucion = crear_solucion(Nodos)
    distancia_solucion = distancia_total(solucion, problem)
    if distancia_solucion < mejor_distancia:
        mejor_solucion = solucion
        mejor_distancia = distancia_solucion

# print("La mejor solucion encontrada es ", end="")
# print(mejor_solucion)
# print("Con una distancia total de ", end="")
# print(mejor_distancia)
# print(mejor_distancia, mejor_solucion )
return mejor_solucion

```

In [8]: `solucion = busqueda_aleatoria(problem, 100)`

```
print("Recocido simulado: ", end="")
print(solucion)
print("Distancia: ", end="")
print(distancia_total(solucion, problem))
```

Recocido simulado: [0, 7, 30, 15, 25, 24, 35, 36, 38, 14, 4, 6, 27, 26, 3, 29, 16, 33, 32, 28, 34, 18, 5, 11, 41, 10, 37, 17, 1, 20, 19, 12, 8, 9, 40, 21, 39, 2, 23, 13, 31, 22]
Distancia: 4098

```
In [0]: def generar_vecina(solucion):
    mejor_solucion = []
    mejor_distancia = 10e10
    for i in range(1, len(solucion)-1):
        for j in range(i+1, len(solucion)):
            vecina = solucion[:i] + [solucion[j]] + solucion[i+1:j] + [solucion[i]] + solucion[j+1:]
            distancia_vecina = distancia_total(vecina, problem)
            if distancia_vecina <= mejor_distancia:
                mejor_distancia = distancia_vecina
                mejor_solucion = vecina
    return mejor_solucion
```

```
In [0]: def busqueda_local(problem, N):
    mejor_solucion = []
    mejor_distancia = 10e100

    Nodos = list(problem.get_nodes())
    solucion_referencia = crear_solucion(Nodos)

    for i in range(N):
        vecina = generar_vecina(solucion_referencia)
        distancia_vecina = distancia_total(vecina, problem)
        if distancia_vecina <= mejor_distancia:
            mejor_distancia = distancia_vecina
            mejor_solucion = vecina

        solucion_referencia = vecina

    #print("La mejor solucion encontrada es ", end="")
    #print(mejor_solucion)
```

```

#print(mejor_solucion)
#print("Con una distancia total de ", end="")
#print(mejor_distancia)
#print(mejor_distancia, mejor_solucion)
return mejor_solucion

```

```

In [11]: solucion = busqueda_local(problem, 1000)
print("Recocido simulado: ", end="")
print(solucion)
print("Distancia: ", end="")
print(distancia_total(solucion, problem))

```

```

Recocido simulado: [0, 1, 3, 2, 27, 31, 17, 36, 35, 20, 33, 34, 32, 30, 10, 25, 11, 12, 1
5, 37, 7, 18, 41, 23, 40, 24, 38, 22, 39, 21, 9, 8, 29, 28, 4, 6, 26, 5, 13, 19, 16, 14]
Distancia: 1765

```

```

In [0]: def probabilidad(T,d):
        r = random.random()
        if r >= (e**(-1*d)/((T*0.5*10**(-5)))):
            return True
        else:
            return False

def bajar_temperatura(T):
    return T*0.9

```

```

In [0]: def recocido_simulado(problem, TEMPERATURA):
        solucion_referencia = crear_solucion(Nodos)
        distancia_referencia = distancia_total(solucion_referencia, problem)

        mejor_solucion = []
        mejor_distancia = 10e100

        while(TEMPERATURA > 1):
            #print("#Temperatura: ", TEMPERATURA, " Fitness: " , mejor_distancia)
            vecina = generar_vecina(solucion_referencia)
            distancia_vecina = distancia_total(vecina, problem)

            if distancia_vecina < mejor_distancia:
                mejor_distancia = distancia_vecina

```



```

        mejor_distancia = distancia_vecina
        mejor_solucion = vecina

        if distancia_vecina < distancia_referencia or probabilidad(TEMPERATURA, abs(distancia_
referencia - distancia_vecina)):
            solucion_referencia = vecina
            distancia_referencia = distancia_vecina

        TEMPERATURA = bajar_temperatura(TEMPERATURA)

        #print(mejor_distancia, mejor_solucion )

    return mejor_solucion

```

```

In [14]: solucion = recocido_simulado(problem, 10000)
print("Recocido simulado: ", end="")
print(solucion)
print("Distancia: ", end="")
print(distancia_total(solucion, problem))

```

Recocido simulado: [0, 1, 10, 25, 11, 12, 18, 26, 2, 27, 3, 4, 6, 5, 13, 19, 16, 36, 35, 2
0, 33, 34, 30, 38, 22, 39, 21, 9, 8, 41, 23, 40, 24, 29, 28, 7, 14, 15, 37, 17, 31, 32]
Distancia: 1681

```

In [0]: def Add_Nodo(problem, H, T):
        Nodos = list(problem.get_nodes())
        return random.choice(list(set(range(1, len(Nodos))) - set(H)))

    def Incrementa_Feromona(problem, T, H):
        for i in range(len(H)-1):
            T[H[i]][H[i+1]] += 1000/distancia_total(H, problem)
        return T

    def Evaporar_Feromonas(T):
        T = [[max(T[i][j] - 0.3, 1) for i in range(len(Nodos)) ] for j in range(len(Nodos)) ]
        return T

    def hormigas(problem, N):
        Nodos = list(problem.get_nodes())
        Aristas = list(problem.get_edges())

```

```

T = [[1 for _ in range(len(Nodos))] for _ in range(len(Nodos))]

Hormiga = [[0] for _ in range(N)]

for h in range(N):
    for i in range(len(Nodos)-1):
        Nuevo_Nodo = Add_Nodo(problem, Hormiga[h], T)
        Hormiga[h].append(Nuevo_Nodo)

    T = Incrementa_Feromona(problem, T, Hormiga[h])

    T = Evaporar_Feromonas(T)

mejor_solucion = []
mejor_distancia = 10e100
for h in range(N):
    distancia_actual = distancia_total(Hormiga[h], problem)
    if distancia_actual < mejor_distancia:
        mejor_solucion = Hormiga[h]
        mejor_distancia = distancia_actual

#print(mejor_distancia, mejor_solucion )
return mejor_solucion

```

```

In [16]: solucion = hormigas(problem, 1000)
print("Colonia hormigas: ", end="")
print(solucion)
print("Distancia: ", end="")
print(distancia_total(solucion, problem))

```

```

Colonia hormigas: [0, 19, 29, 4, 20, 36, 6, 26, 38, 1, 5, 18, 9, 2, 25, 14, 11, 34, 24, 2
1, 40, 28, 13, 10, 8, 22, 32, 16, 37, 3, 7, 12, 39, 23, 30, 41, 33, 35, 17, 31, 15, 27]
Distancia: 3967

```

In [0]:

