Weighted Interval Scheduling

February 7, 2024

[1]:
```
import random

def random_request():
    return [sorted(random.sample(range(100),2)), random.random()*10]
```

[8]:
```
R = random_request()
print(R[0])
print(R[1])
```

```
[7, 87]
2.965377096842471
```

[3]:
```
def make_requests(n):
    return [random_request() for i in range(n)]
```

[4]:
```
make_requests(3)
```

[4]:
```
[[[17, 31], 0.8251616128916606],
 [[67, 77], 6.909995125840958],
 [[33, 78], 9.161103502191734]]
```

[5]:
```
def compatible(r1, r2):
    return r2[0][1] <= r1[0][0] or r2[0][0] >= r1[0][1]

def is_compatible(request, solution):
    return all(compatible(request, r) for r in solution)
```

[6]:
```
def plot_requests(requests):
    for r in sorted(requests, key=lambda x : x[0][1]):
        print(" "*(r[0][0]) + "-"*(r[0][1]-r[0][0]) + "  (" +␣
 ↪str(round(r[1],2)) + ")")
    #print("total value:",sum(r[1] for r in requests))
    total_value = sum(r[1] for r in requests)
    print(f"total value: {total_value}")
```

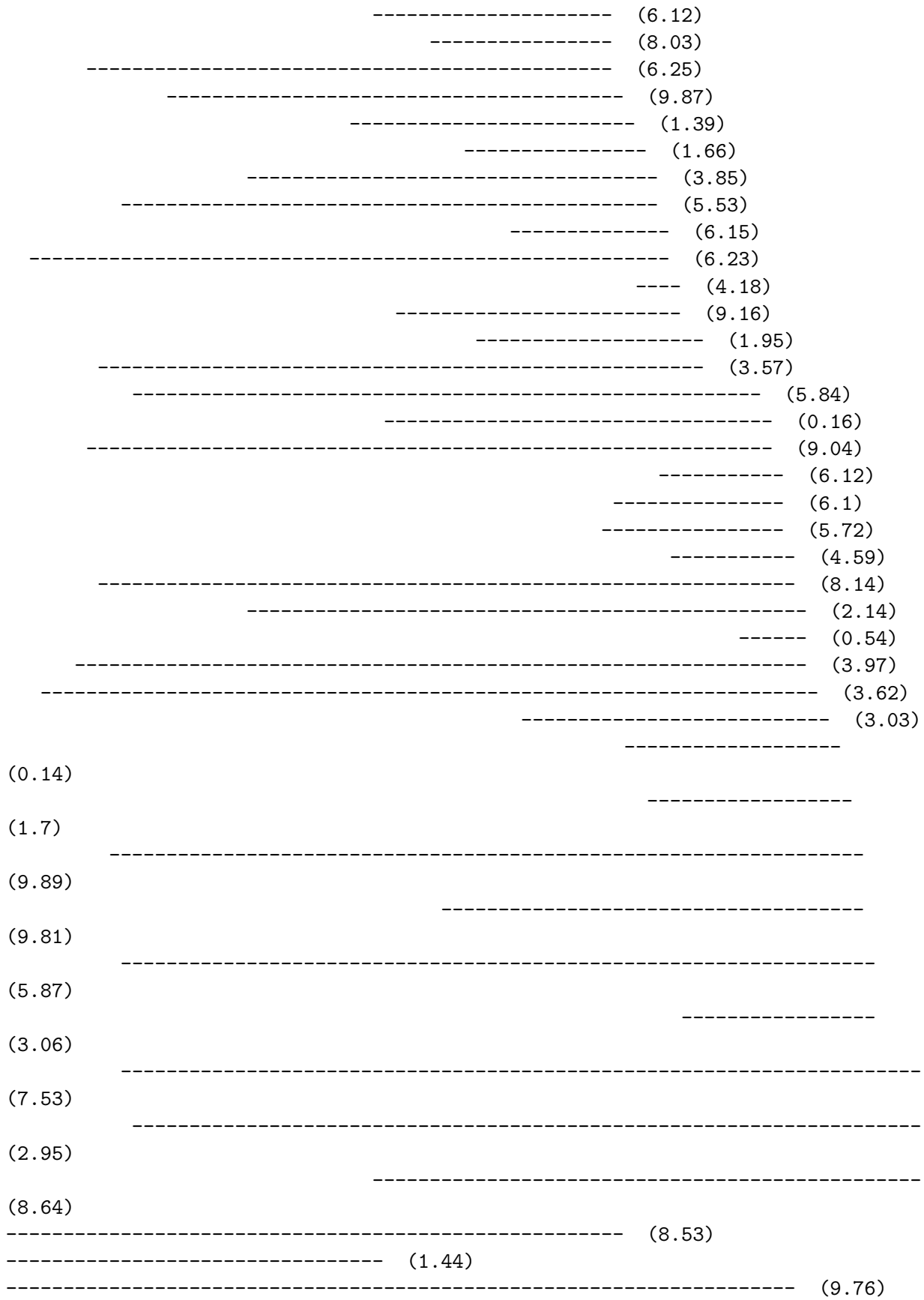
[9]:
```
# best = most valuable
def greedy(requests):
    sorted_requests = sorted(requests, key=lambda r: r[1], reverse=True)
```

```
    solution = []
    solution.append(sorted_requests.pop(0))

    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        if is_compatible(request, solution):
            solution.append(request)

    return solution
```

[10]:
```
requests = make_requests(100)
```

[11]:
```
plot_requests(requests)
```

```
      --  (3.93)
          ----  (9.17)
     -------------  (1.41)
             ----  (4.01)
                 -  (0.3)
    -------------------  (9.2)
                -------  (7.94)
-------------------------  (7.78)
                    ---  (6.91)
                 ----------  (1.54)
    ---------------------------  (1.24)
                           ---  (4.76)
                           -----  (0.26)
                     --------------  (7.73)
                       ------------  (9.32)
      ------------------------------------  (3.95)
-----------------------------------------  (6.19)
              ----------------------------  (0.81)
                     --------------------  (7.63)
                                        --  (1.76)
  ------------------------------------------  (9.05)
                                      ----  (9.93)
                                        ---  (6.69)
                                 -------------  (5.06)
                               ---------------  (9.09)
                            --------------------  (1.65)
                        -------------------------  (3.19)
                                         ---  (5.95)
                                    -----------  (6.0)
                        --------------------------  (5.65)
 ------------------------------------------------  (8.88)
                                    -------------  (7.93)
                                 -----------------  (1.84)
                                 ------------------  (9.18)
```



```
------------------------------------------------  (8.74)
-------------------------------------------------------------  (3.15)
-----  (3.89)
-----------------------------------------------------------  (4.76)
---------------------------------------  (0.47)
------------------------------  (0.15)
-----------------  (2.27)
-----------------------------------------------------------  (6.85)
-------------------------------------------------  (4.71)
--------------------------------  (7.88)
-----------------------------------  (1.23)
------------------------------------------------------------------------
(4.51)
-----------------------------------------------------------  (1.72)
-----------------------------------------------------------  (0.35)
-----------  (2.22)
-----------------------------------  (5.67)
-------------------------------------------------------------  (2.14)
-------------  (3.0)
          ---  (2.87)
-----------------------------------------------------------------------  (3.11)
-------------------------------------------------------------  (3.86)
         ------------------------------------------------------------------
-----------------  (1.78)
------------------------------------------------  (5.45)
-----------------------------------------------------------------------  (0.75)
               ---------------------------------------------------------------
------------------  (6.74)
              ------------------------------------------------------------------
------------------  (1.3)
------------------------------------------------  (4.86)
total value: 482.5960340164662
```

[12]:
```
sol = greedy(requests)
print(sol)
print(len(sol))
```

```
[[[43, 47], 9.925179405187292], [[25, 37], 9.316275686124113], [[5, 24],
9.199936953280996], [[54, 87], 7.878205430012929], [[90, 93],
2.8664237057941158]]
5
```

[13]:
```
plot_requests(sol)
```

```
     -------------------  (9.2)
                        ------------  (9.32)
                                    ----  (9.93)
--------------------------------  (7.88)
          ---  (2.87)
```

```
total value: 39.18602118039944
```

[ ]:
```
```

[ ]:
```
sum(s[1] for s in sol)
```

[ ]:
```
# best = most valuable
# best = shortest
# best = most value-dense (highest value/duration)
```

[16]:
```
def greedy(requests, sort_function):
    sorted_requests = sorted(requests, key=sort_function)
    solution = []
    solution.append(sorted_requests.pop(0))

    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        if is_compatible(request, solution):
            solution.append(request)

    return solution
```

[ ]:
```
[2, 9, 1]
[-2, -9, -1]
[-9, -2, -1]
[9, 2, 1]
```

[17]:
```
# request = [[start, end], value]
most_value = lambda req : -req[1]
shortest = lambda req : req[0][1] - req[0][0]
density = lambda req : -req[1]/(req[0][1] - req[0][0])
```

[ ]:
```
print(most_value)
print(most_value([[10,20], 15.1]))
```

[18]:
```
requests = make_requests(1000)
```

[19]:
```
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
s3 = greedy(requests, density)
```

[20]:
```
plot_requests(s1)
```

```
--  (6.6)
   -  (6.96)
     --  (2.49)
       ---  (8.96)
          -  (2.27)
------------------------------------------------------------------------  (9.99)
```

```
       ---  (9.47)
           ----  (9.02)
total value: 55.77038811227591
```

[21]:
```
plot_requests(s2)
```

```
--  (6.6)
   -  (6.96)
     --  (2.49)
       -  (8.11)
          -  (2.27)
           ---  (3.28)
               -  (4.32)
                  --  (9.22)
                     -  (7.47)
                       --  (1.73)
                         ---  (0.36)
                            -  (1.88)
                             -  (3.29)
                              -  (8.84)
                                -  (2.03)
                                 --  (8.99)
                                       -  (2.06)
                                         -  (2.59)
                                           --  (7.01)
                                             -  (7.55)
                                              -  (8.85)
                                               -  (1.71)
                                                -  (8.6)
                                                   -  (6.19)
                                                    ---  (4.63)
                                                        -  (9.2)
                                                             ---  (0.25)
                                                                -  (0.18)
                                                                    -  (2.12)
                                                                      ---
(8.36)
                                                                          --
(2.22)
                                                                            --
(3.96)
                                                                              -
(0.63)
---  (4.42)
       ---  (9.47)
           --  (7.68)
total value: 175.48607027063144
```

[22]:
```
plot_requests(s3)
```

```
--  (6.6)
   -  (6.96)
     --  (2.49)
       -  (8.11)
         ---  (8.95)
               -  (4.32)
                  --  (9.22)
                     -  (7.47)
                       --  (1.73)
                         ---  (0.36)
                            -  (1.88)
                             -  (3.29)
                              -  (8.84)
                                -  (7.51)
                                 --  (8.99)
                                      ---  (7.79)
                                         -  (2.59)
                                           --  (7.01)
                                             -  (7.55)
                                              -  (8.85)
                                               -  (1.71)
                                                -  (8.6)
                                                   -  (6.19)
                                                    ----  (8.3)
                                                        -  (9.2)
                                                         -------  (8.78)
                                                                ---  (6.46)
                                                                    -  (2.12)
                                                                      ---
(8.36)                                                                    --
(2.22)                                                                      --
(3.96)                                                                        -
(0.63)
---  (5.49)
       ---  (9.47)
           --  (7.68)
total value: 209.6570782815415
```

[ ]:

[29]:
```
requests = make_requests(100_000)
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
```

```
s3 = greedy(requests, density)
def score(sol):
    return sum(s[1] for s in sol)
print([score(s1), score(s2), score(s3)])
```

```
[158.27007914208852, 493.65631072775165, 940.3412225719388]
```

[ ]:

[ ]: