Weighted Interval Scheduling

February 6, 2023

[2]:
```
import random

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

[6]:
```
random_request()
```

[6]:
```
[[4, 77], 7.812850208565464]
```

[7]:
```
def make_requests(n):
    return [random_request() for i in range(n)]
```

[8]:
```
make_requests(3)
```

[8]:
```
[[[19, 93], 4.544893361987881],
 [[22, 32], 8.936559198028235],
 [[8, 33], 6.972678462477631]]
```

[9]:
```
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)
```

[10]:
```
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}")
```

[11]:
```
# best = most valuable
def greedy(requests):
    sorted_requests = sorted(requests, key=lambda r: r[1], reverse=True)
    solution = []
    solution.append(sorted_requests.pop(0))
```

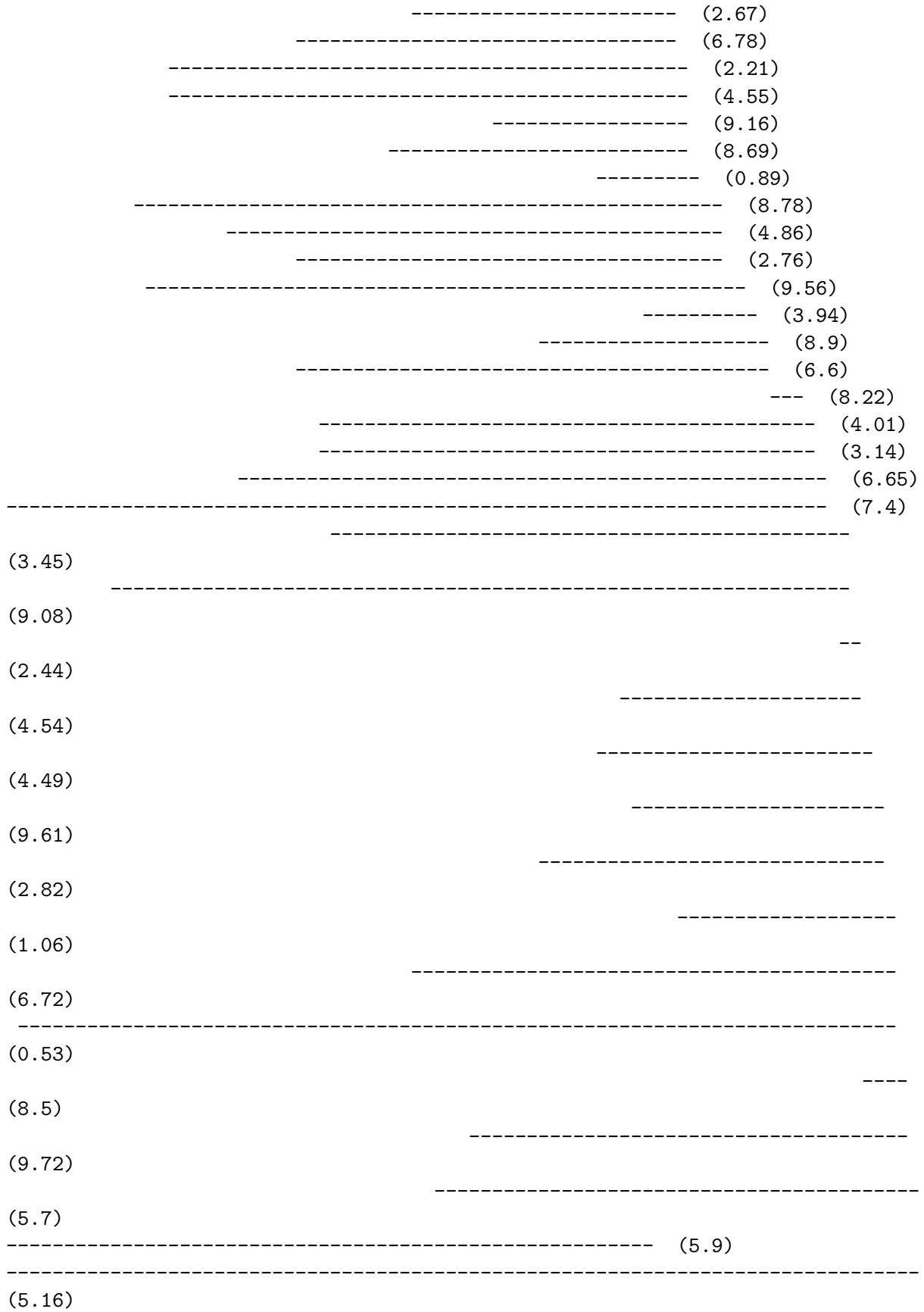
```
    while len(sorted_requests) > 0:
        request = sorted_requests.pop(0)
        if is_compatible(request, solution):
            solution.append(request)

    return solution
```

[15]:
```
requests = make_requests(100)
```

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

```
  -----------------  (7.82)
  ---------------------  (2.03)
            ----------  (7.07)
              ---------  (8.35)
               ----------  (1.01)
     ----------------------  (8.02)
                 -----------  (8.88)
--------------------------------  (6.5)
            ----------------------  (9.51)
             --------------------  (3.25)
         --------------------------  (0.6)
                 -------------------  (8.45)
                 --------------------  (8.01)
           ---------------------------  (7.57)
 -----------------------------------  (7.03)
                          ------------  (7.42)
                      ----------------  (9.32)
                               -------  (2.18)
                 ----------------------  (1.67)
                          -------------  (2.67)
      --------------------------------  (4.57)
        ---------------------------------  (6.46)
                             ----------  (6.24)
                              ----------  (0.7)
                          ----------------  (7.98)
                           ----------------  (4.37)
    ----------------------------------------  (8.93)
                             --------------  (4.85)
                     -------------------------  (1.24)
                 -------------------------------  (5.47)
                                    --------  (2.27)
                           ---------------------  (5.19)
        ------------------------------------------  (0.31)
                                  ------------------  (9.79)
                                    ----------------  (7.75)
            ----------------------------------------  (9.45)
          ----------------------------------------------  (5.05)
```



```
---------------------------------------  (0.48)
-----------------------------------  (2.53)
-------------------------  (8.03)
-----------------------------------------------------------  (3.84)
-------------------------------------------------------  (4.65)
-------------------------------------------  (5.43)
-----------------------------------  (4.14)
----------------------------------------  (5.24)
----------------  (3.85)
--------------  (7.28)
-------------------------------------  (9.58)
------------------  (8.91)
----------------  (7.95)
-----------------------------------------  (3.41)
-----------------------------------  (3.72)
-------------  (9.45)
-----------------------------------------------  (6.11)
-----------------------------------------  (7.84)
------------------------------------------------------------------------  (1.81)
     ---------------------------------------------------------------------------
---------  (5.73)
           -  (2.65)
-------------------------------------------------------------------------------
(6.8)
    ----------------------------------------------------------------------------
--------------  (5.71)
-------------------------------------------------------------------------
(1.32)
--------------------  (1.37)
-----------------------------------------------------------------------
(2.09)
----------------------------------  (6.11)
----------------  (5.9)
-----------------------------------------------------------  (1.78)
total value: 541.1882013681587
```

[17]:
```
sol = greedy(requests)
print(sol)
print(len(sol))
```

```
[[[0, 17], 9.642055670118738], [[33, 57], 9.584198092734097], [[68, 96],
9.402787196546646], [[19, 28], 4.107258566760387]]
4
```

[18]:
```
plot_requests(sol)
```

```
----------------  (9.64)
                  ---------  (4.11)
                             ------------------------  (9.58)
```

```
---------------------------  (9.4)
total value: 32.73629952615987
```

[ ]:

[19]:
```
sum(s[1] for s in sol)
```

[19]:
```
32.73629952615987
```

[14]:
```
# best = most valuable
# best = shortest
# best = most value-dense (highest value/duration)
```

[20]:
```
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
```

[21]:
```
# 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])
```

[23]:
```
print(most_value)
print(most_value([[10,20], 15.1]))
```

```
<function <lambda> at 0x10b933060>
-15.1
```

[24]:
```
requests = make_requests(1000)
```

[25]:
```
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
s3 = greedy(requests, density)
```

[27]:
```
plot_requests(s1)
```

```
--------  (9.55)
        ---  (3.05)
           --  (8.3)
              --------  (9.95)
                        ----  (9.57)
```

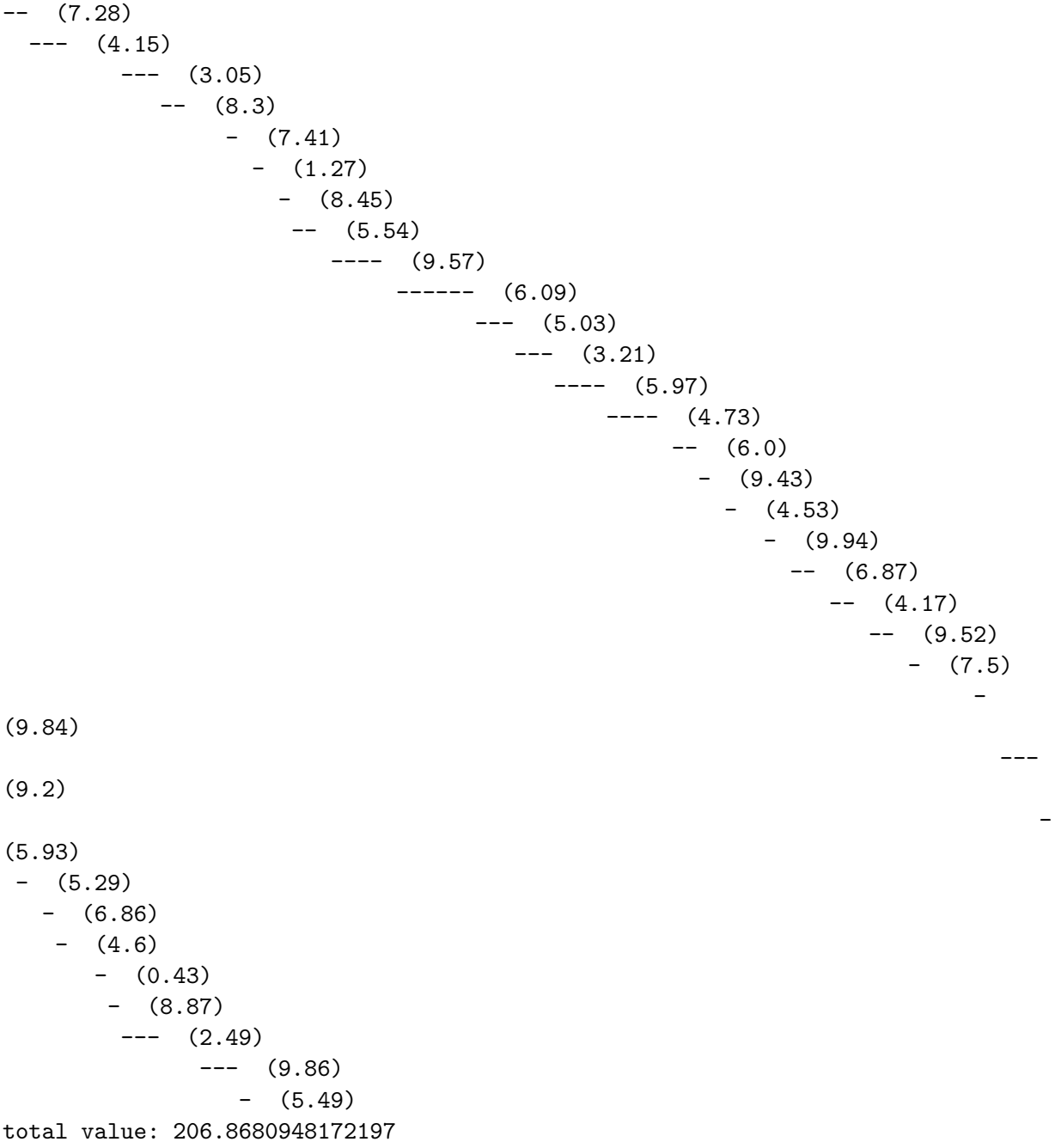
```
                                  ---------------------------------------  (9.98)
                                                                           -
(9.84)
                                                                             ---
(9.2)
------  (7.55)
      -------  (9.96)
               ---  (9.86)
                  -  (5.49)
total value: 102.30311439286716
```

[28]:
```
plot_requests(s2)
```

```
 --  (2.65)
    --  (2.31)
         ---  (3.05)
            --  (8.3)
                 -  (7.41)
                   -  (1.27)
                     -  (8.45)
                      --  (5.54)
                        --  (1.08)
                            -----  (5.07)
                                   ---  (5.03)
                                        -  (0.87)
                                         --  (1.08)
                                              --  (0.55)
                                                  --  (6.0)
                                                    -  (9.43)
                                                      -  (4.53)
                                                         -  (9.94)
                                                            -  (2.16)
                                                             --  (2.11)
                                                                 --  (9.52)
                                                                    -  (7.5)
                                                                       -
(9.84)
                                                                         --
(4.36)
                                                                            -
(5.93)
 -  (5.29)
   -  (6.86)
    -  (4.6)
       -  (0.43)
        -  (8.87)
         ---  (2.49)
             ---  (2.49)
```

```
                  -  (0.22)
                   -  (5.49)
total value: 160.731839170667
```

[29]:
```
plot_requests(s3)
```




```
total value: 206.8680948172197
```

[ ]:
```

```

requests = make_requests(1000)

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)])

[ ]: