Weighted Interval Scheduling

January 30, 2026

[1]:
```
import random

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

[5]:
```
R = random_request()
print(R)
print(R[0])
print(R[1])
```

```
[[27, 33], 4.259970557121243]
[27, 33]
4.259970557121243
```

[6]:
```
def make_requests(n):
    return [random_request() for i in range(n)]
```

[7]:
```
make_requests(3)
```

[7]:
```
[[[14, 63], 3.813388978956583],
 [[40, 43], 8.626891962979318],
 [[10, 76], 0.44221148299870894]]
```

[8]:
```
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)
```

[9]:
```
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}")
```

[10]:
```
R
```

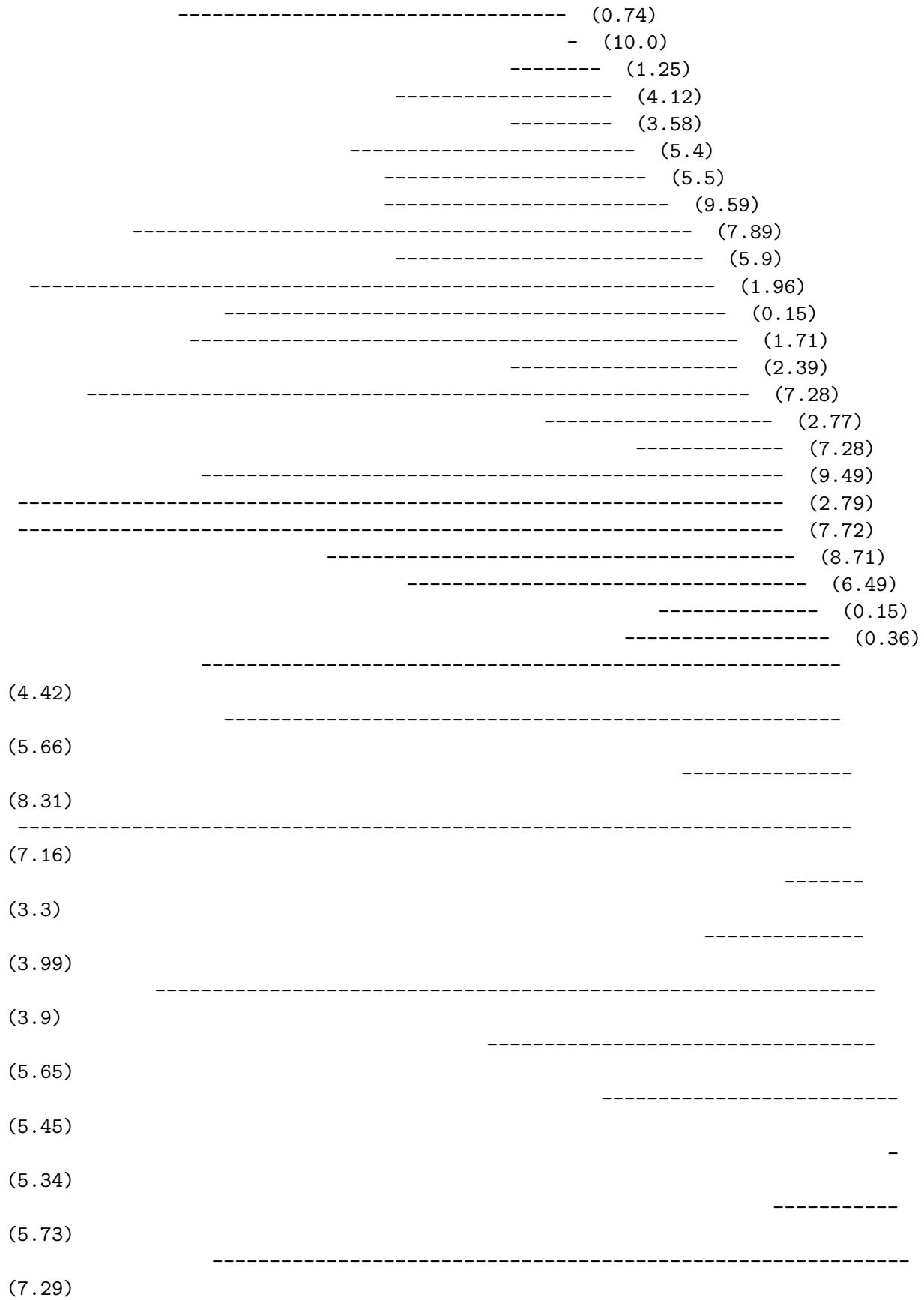
[10]:
```
[[27, 33], 4.259970557121243]
```

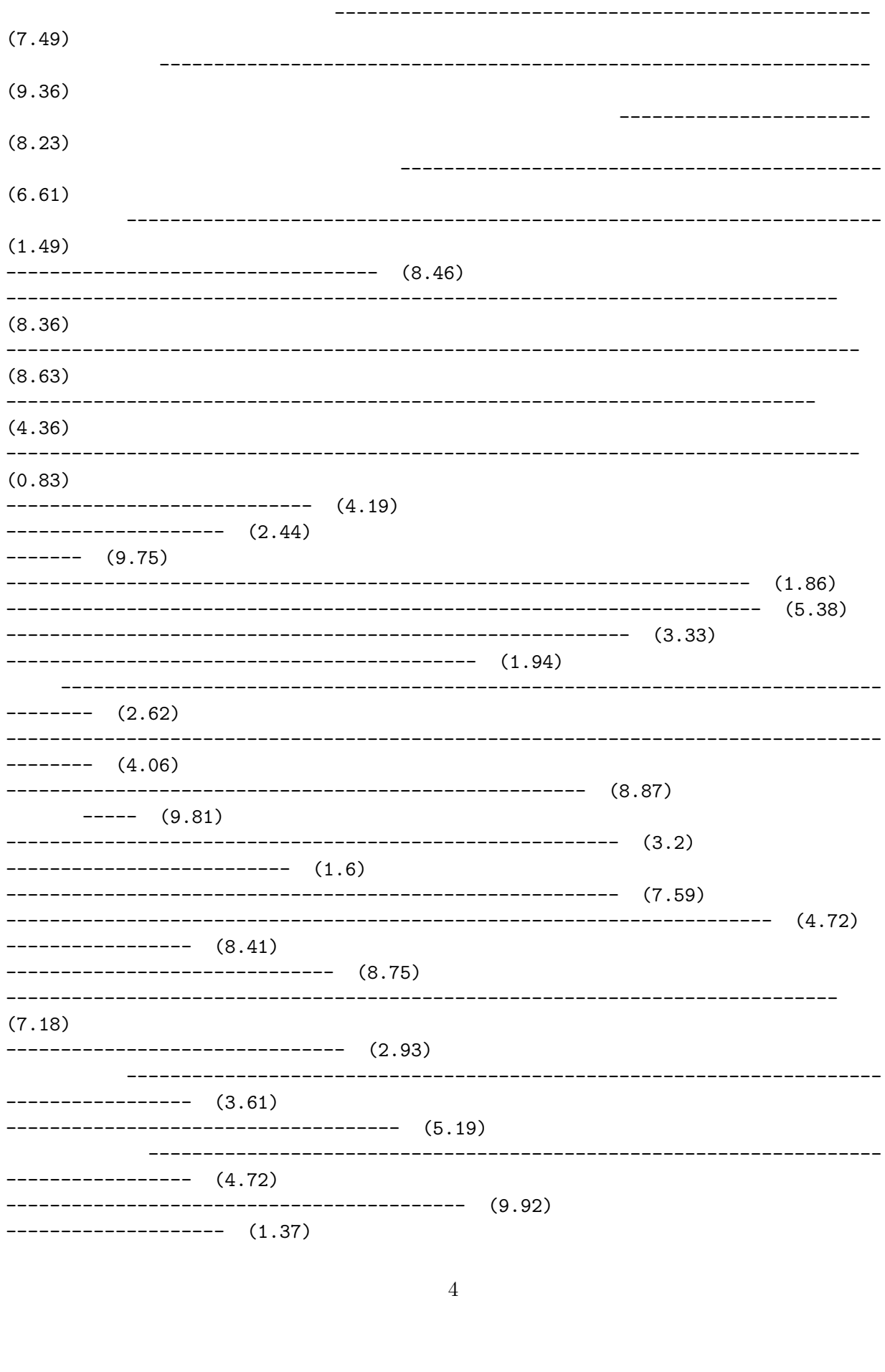
[11]:
```
# best = most valuable
def greedy(requests):
    sorted_requests = sorted(requests, key=lambda r: r[1], reverse=True)
    # also could have done
    # sorted_requests = sorted(requests, key=lambda r: -r[1])
    solution = []
    solution.append(sorted_requests.pop(0))

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

    return solution
```

[12]:
```
requests = make_requests(100)
```

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

```
    --------  (2.53)
       -----  (1.0)
 --------------  (9.29)
             --  (4.52)
   -----------------  (5.77)
            ---------  (9.54)
     ----------------  (1.34)
                    -  (3.19)
       ----------------  (5.69)
        ----------------  (3.18)
                    ----  (6.7)
  -------------------------  (8.45)
       -------------------------  (3.86)
                          ---------  (4.76)
 ----------------------------------  (1.76)
  ---------------------------------  (0.86)
                        --------------  (8.81)
                     -----------------  (0.51)
                      -----------------  (7.69)
       ----------------------------------  (0.58)
   ----------------------------------  (5.14)
                     -------------------  (8.3)
                             ------------  (7.39)
              ------------------------------  (8.47)
                          ------------------  (6.48)
                                    --------  (1.6)
                                 ------------  (6.2)
```





```
----------------------------------------  (3.33)
-------------------------  (3.37)
------------------------------------------------------------------------  (5.13)
total value: 512.0885879560853
```

[14]:
```
sol = greedy(requests)
print(sol)
print(len(sol))
```

```
[[[49, 50], 9.999411125110324], [[55, 97], 9.92176595875139], [[13, 22],
9.53960402648078], [[26, 40], 8.806833726549026], [[23, 24], 3.189200897465505],
[[5, 13], 2.5295233838369358]]
6
```

[15]:
```
plot_requests(sol)
```

```
     --------  (2.53)
            ---------  (9.54)
                     -  (3.19)
                        --------------  (8.81)
                                              -  (10.0)
----------------------------------------  (9.92)
total value: 43.98633911819396
```

[16]:
```
sol
```

[16]:
```
[[[49, 50], 9.999411125110324],
 [[55, 97], 9.92176595875139],
 [[13, 22], 9.53960402648078],
 [[26, 40], 8.806833726549026],
 [[23, 24], 3.189200897465505],
 [[5, 13], 2.5295233838369358]]
```

[17]:
```
sum(s[1] for s in sol)
```

[17]:
```
43.98633911819396
```

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

[18]:
```
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
```

[ ]:

[19]:
```
# 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])
```

[ ]:

[25]:
```
requests = make_requests(1000)
```

[26]:
```
s1 = greedy(requests, most_value)
s2 = greedy(requests, shortest)
s3 = greedy(requests, density)
```

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

```
 ---  (7.87)
     -------  (6.76)
            --------------------------  (9.99)
                                     -  (5.76)
                                      ---------------------------------  (9.77)
---------------  (9.97)
        ------  (9.67)
              --  (9.83)
                  -  (8.92)
total value: 78.54253900972864
```

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

```
  -  (4.5)
     --  (1.91)
       ---  (4.17)
          --  (0.46)
               -  (3.64)
                --  (6.98)
                    -  (2.0)
                     ------  (1.64)
                           --  (3.3)
                             -  (9.16)
                               --  (4.1)
                                 --  (7.66)
                                   ---  (4.38)
                                      -  (5.76)
```

```
                                                  -  (5.71)
                                                     --  (1.23)
                                                       -  (4.46)
                                                          -  (0.33)
                                                            -  (8.68)
                                                               ---  (6.21)
                                                                  -  (8.87)
                                                                    -  (9.06)
                                                                       --  (2.86)
                                                                            --  (1.35)
                                                                                 --
(4.21)
                                                                                    -
(0.98)
                                                                                     -
(2.91)
                                                                                       -
(6.47)
--  (9.36)
     ---  (0.67)
         ---  (5.9)
            --  (6.86)
              --  (5.81)
                  -  (8.92)
total value: 160.50441654524232
```

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

```
  -  (4.5)
   ---  (1.03)
      --  (2.0)
         ------  (9.06)
               -  (3.64)
                --  (6.98)
                  ----  (9.88)
                           --  (3.3)
                             -  (9.16)
                               --  (4.1)
                                 --  (7.66)
                                   ---  (4.38)
                                      -  (5.76)
                                         -  (5.71)
                                           ---  (8.56)
                                              -  (4.46)
                                                 -  (0.33)
                                                   -  (8.68)
                                                      ---  (6.21)
                                                         -  (8.87)
```

```
                                                                  -  (9.06)
                                                                    ---  (1.99)
                                                                       ---  (6.8)
                                                                          --  (1.35)
                                                                              --
(4.21)
                                                                                 -
(0.98)
                                                                                  -
(2.91)
                                                                                    -
(6.47)
--  (9.36)
    ----  (2.88)
         ---  (5.9)
            --  (6.86)
              --  (9.83)
                  -  (8.92)
total value: 191.78616631663735
```

[ ]:

[30]:
```
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)])
```

```
[242.86556697353143, 471.9048622334046, 932.721185235033]
```

[ ]:

[ ]: