

Weighted Interval Scheduling

February 14, 2022

```
[1]: import random

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

```
[3]: random_request()
```

```
[3]: [[38, 54], 4.1886485190133635]
```

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

```
[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)
```

```
[24]: def plot_requests(requests):
    for r in sorted(requests, key=lambda x : x[0][1]):
        print(" *{}*({r[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}")
```

```
[7]: # 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
```

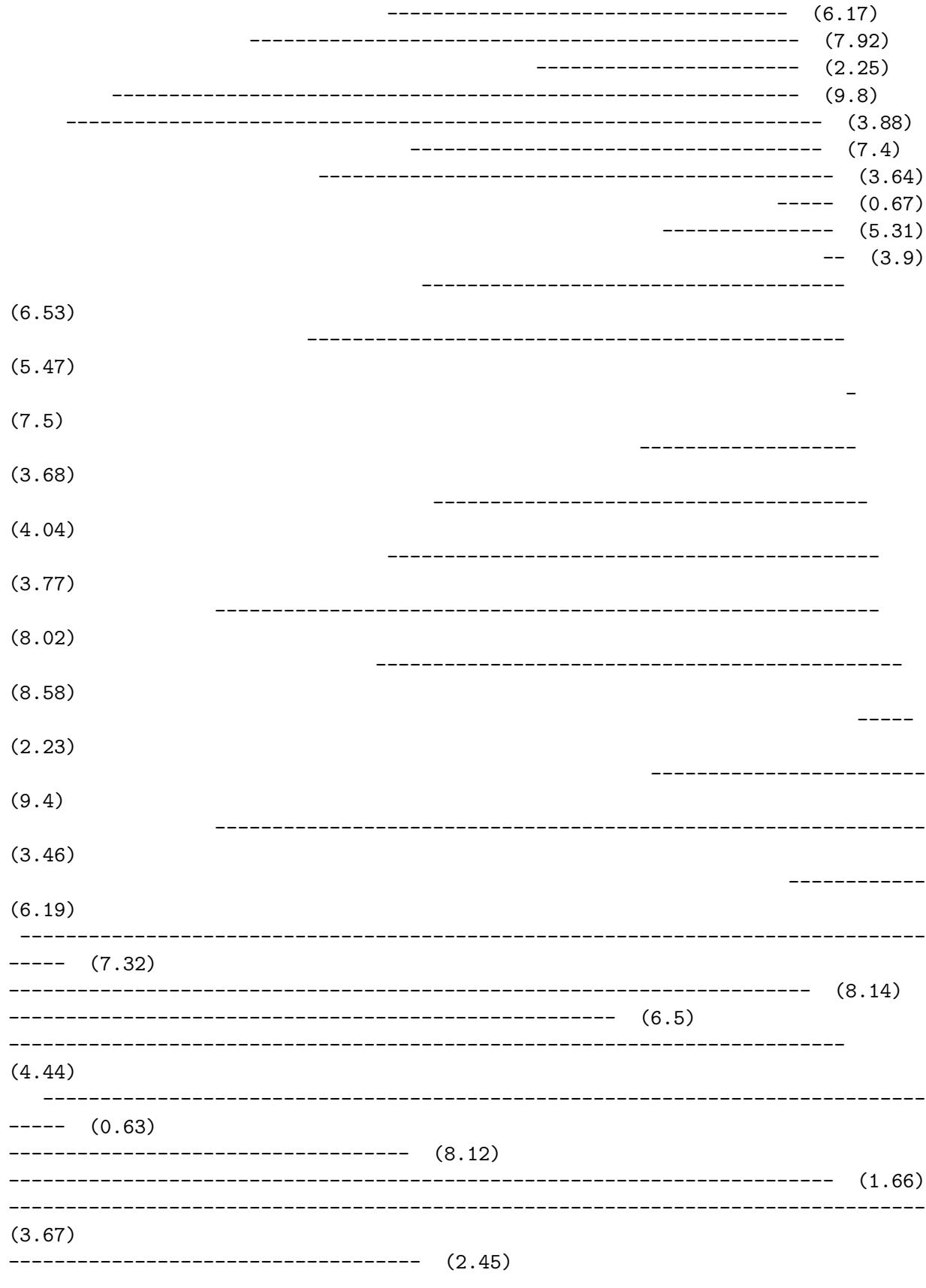
```
[8]: requests = make_requests(100)
```

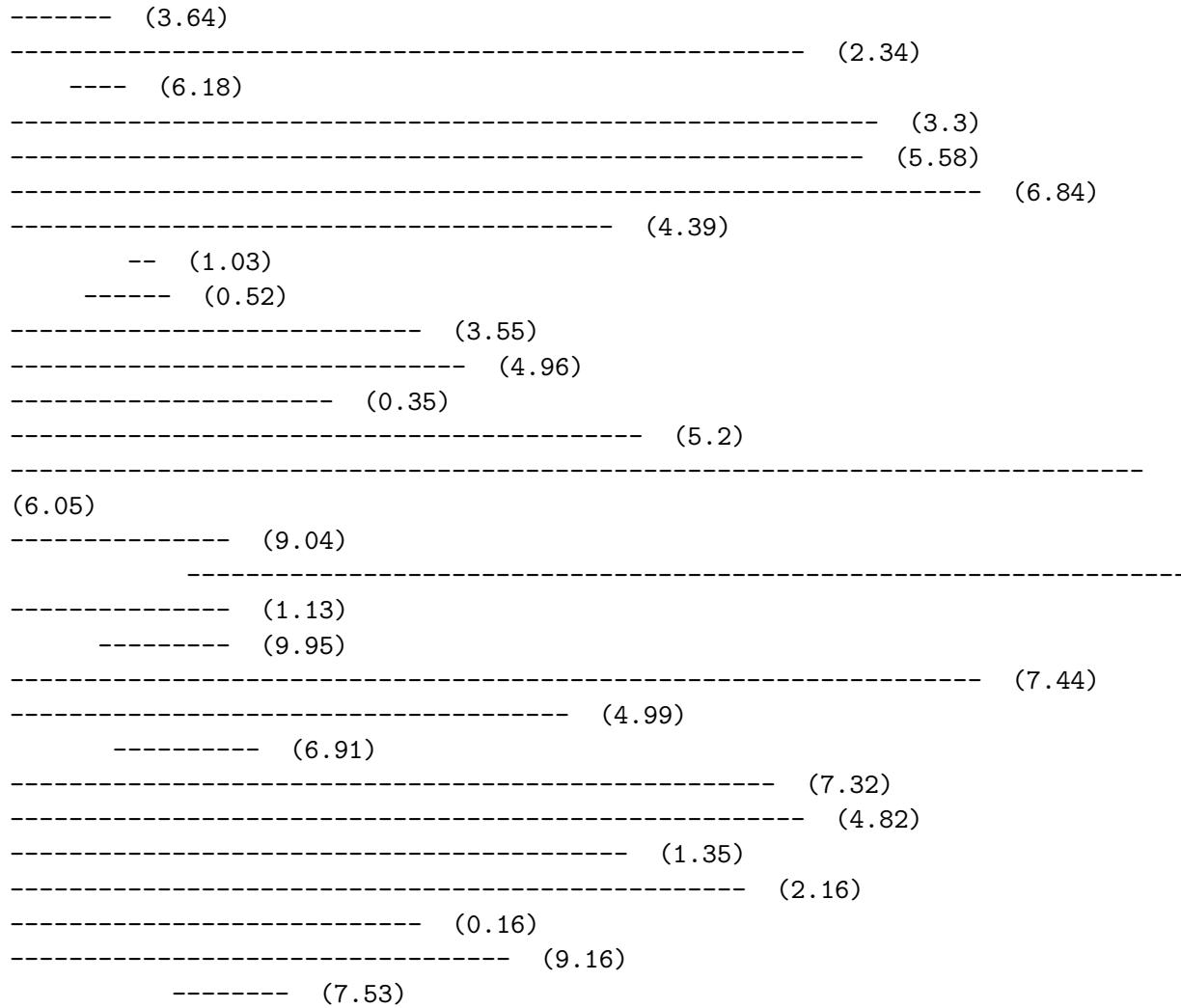
```
[9]: sol = greedy(requests)
```

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

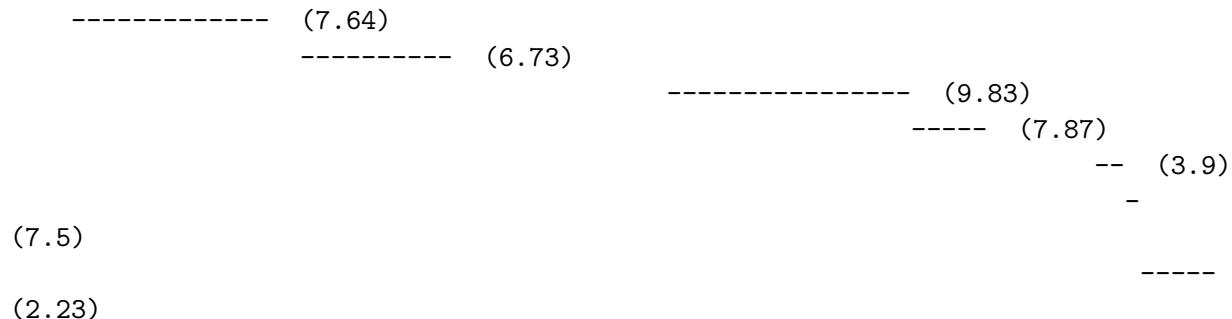
The visualization shows 100 requests plotted as horizontal dashed lines. Each line represents a request's duration, starting at its start time and ending at its end time. The labels indicate the start and end times for each request.

Request	Start Time	End Time
1	(7.06)	(2.8)
2	(5.05)	(7.64)
3	(2.12)	(6.73)
4	(6.56)	(5.47)
5	(1.36)	(1.19)
6	(6.97)	(1.85)
7	(2.55)	(2.07)
8	(9.56)	(9.05)
9	(9.34)	(9.24)
10	(5.96)	(5.45)
11	(5.81)	(3.42)
12	(4.93)	(9.43)
13	(9.43)	(7.44)
14	(6.06)	(7.28)
15	(7.28)	(9.83)
16	(9.8)	(0.43)
17	(9.31)	(5.9)
18	(5.9)	(7.07)
19	(7.07)	(1.77)
20	(1.77)	(7.87)
21	(7.87)	(1.5)
22	(1.5)	(5.51)
23	(5.51)	(8.28)
24	(8.28)	(8.65)
25	(8.65)	(7.6)
26	(7.6)	(6.48)
27	(6.48)	(8.43)





[11]: `plot_requests(sol)`



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

[13]: 55.64068759685977

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

[15]: 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

[23]: # 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])

[17]: print(most_value)

<function <lambda> at 0x1044bf4c0>

[30]: requests = make_requests(1000)

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

[32]: plot_requests(s1)

-- (8.41)
-- (7.99)
- (9.88)
- (3.24)
- (7.75)
- (4.72)
---- (8.9)
----- (10.0)
---- (1.84)
- (6.33)
-- (9.82)
---
(3.63)
---
(9.54)
```

```
(9.65)
-----
(9.98)
total value: 111.67204639509534
```

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

```
-- (8.41)
- (2.89)
- (9.88)
- (3.24)
- (7.75)
- (4.72)
-- (1.68)
- (6.77)
- (0.3)
- (6.84)
-- (8.33)
-- (4.39)
-- (6.73)
----- (9.13)
- (8.56)
- (9.34)
- (4.27)
-- (9.23)
-- (7.14)
- (8.35)
- (4.04)
- (1.87)
- (4.08)
----- (1.84)
- (6.33)
-- (9.82)
- (1.26)
-
(6.97)
-
(5.18)
-
(9.65)
-
(2.99)
-- (9.72)
- (4.32)
-- (2.83)
- (2.17)
-- (0.51)
- (6.08)
```

- (7.57)
total value: 215.16286366215562

[34] : plot_requests(s3)

```
-- (8.41)
-- (7.99)
- (9.88)
- (3.24)
- (7.75)
- (4.72)
---- (8.9)
- (6.77)
- (0.3)
- (6.84)
-- (8.33)
-- (4.39)
-- (6.73)
---- (9.13)
- (8.56)
- (9.34)
- (4.27)
-- (9.23)
-- (7.14)
- (8.35)
- (4.04)
- (1.87)
- (4.08)
---- (1.84)
- (6.33)
-- (9.82)
- (1.26)
-
(6.97)
-
(5.18)
-
(9.65)
-
(2.99)
-- (9.72)
- (4.32)
-- (2.83)
- (2.17)
---- (7.32)
- (6.08)
- (7.57)
total value: 234.29789829029843
```

[]:

[]: