Лабораторная работа №2. Задача о рюкзаке 0-1

Илья Седунов,
Вадим Альперович,
17ПМИ

Knapsack 01

- capacity: *165*
- optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`
- optimal weight: *165*, and profit: *309*
  BruteForce optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`

  ```
  optimal weight: 165, and profit **309**
  ```

  Greedy optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`

  ```
  optimal weight: 165, and profit **309**
  ```

  Branch-And-Bound optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`

  ```
  optimal weight: 165, and profit **309**
  ```

  Dynamic optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`

  ```
  optimal weight: 165, and profit **309**
  ```

  Genetic optimal solution: `[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]`

  ```
  optimal weight: 165, and profit **309**
  ```

Knapsack 02

- capacity: *26*
- optimal solution: `[0, 1, 1, 1, 0]`
- optimal weight: *26*, and profit: *51*
  BruteForce optimal solution: `[0, 1, 1, 1, 0]`

  ```
  optimal weight: 26, and profit **51**
  ```

  Greedy optimal solution: `[1, 0, 1, 0, 0]`

  ```
  optimal weight: 23, and profit **47**
  ```

  Branch-And-Bound optimal solution: `[0, 1, 1, 1, 0]`

  ```
  optimal weight: 26, and profit **51**
  ```

  Dynamic optimal solution: `[0, 1, 1, 1, 0]`

  ```
  optimal weight: 26, and profit **51**
  ```

  Genetic optimal solution: `[0, 1, 1, 1, 0]`

  ```
  optimal weight: 26, and profit **51**
  ```

Knapsack 03

- capacity: *190*
- optimal solution: `[1, 1, 0, 0, 1, 0]`
- optimal weight: *190*, and profit: *150*
  BruteForce optimal solution: `[1, 1, 0, 0, 1, 0]`

  ```
  optimal weight: 190, and profit **150**
  ```

  Greedy optimal solution: `[1, 1, 0, 1, 0, 0]`

  ```
  optimal weight: 179, and profit **146**
  ```

  Branch-And-Bound optimal solution: `[1, 1, 0, 0, 1, 0]`

  ```
  optimal weight: 190, and profit **150**
  ```

  Dynamic optimal solution: `[1, 1, 0, 0, 1, 0]`

  ```
  optimal weight: 190, and profit **150**
  ```

  Genetic optimal solution: `[1, 0, 1, 0, 0, 1]`

  ```
  optimal weight: 153, and profit **119**
  ```

Knapsack 04

- capacity: *50*
- optimal solution: `[1, 0, 0, 1, 0, 0, 0]`
- optimal weight: *50*, and profit: *107*
  BruteForce optimal solution: `[1, 0, 0, 1, 0, 0, 0]`

  ```
  optimal weight: 50, and profit **107**
  ```

  Greedy optimal solution: `[1, 1, 0, 0, 1, 1, 0]`

  ```
  optimal weight: 48, and profit **102**
  ```

  Branch-And-Bound optimal solution: `[1, 0, 0, 1, 0, 0, 0]`

  ```
  optimal weight: 50, and profit **107**
  ```

  Dynamic optimal solution: `[1, 0, 0, 1, 0, 0, 0]`

  ```
  optimal weight: 50, and profit **107**
  ```

  Genetic optimal solution: `[1, 1, 0, 0, 0, 1, 1]`

  ```
  optimal weight: 50, and profit **105**
  ```

Knapsack 05

- capacity: *104*
- optimal solution: `[1, 0, 1, 1, 1, 0, 1, 1]`
- optimal weight: *104*, and profit: *900*
  BruteForce optimal solution: `[1, 0, 1, 1, 1, 0, 1, 1]`

optimal weight: 104, and profit **900**

Greedy optimal solution: `[1, 1, 0, 1, 1, 1, 1, 1]`

optimal weight: 97, and profit **858**

Branch-And-Bound optimal solution: `[1, 0, 1, 1, 1, 0, 1, 1]`

optimal weight: 104, and profit **900**

Dynamic optimal solution: `[1, 0, 1, 1, 1, 0, 1, 1]`

optimal weight: 104, and profit **900**

Genetic optimal solution: `[1, 0, 1, 1, 1, 0, 1, 1]`

optimal weight: 104, and profit **900**

Knapsack 06

- capacity: *170*
- optimal solution: `[0, 1, 0, 1, 0, 0, 1]`
- optimal weight: *169*, and profit: *1735*
  BruteForce optimal solution: `[0, 1, 0, 1, 0, 0, 1]`

  optimal weight: 169, and profit **1735**

  Greedy optimal solution: `[1, 1, 1, 0, 0, 0, 0]`

  optimal weight: 140, and profit **1478**

  Branch-And-Bound optimal solution: `[0, 1, 0, 1, 0, 0, 1]`

  optimal weight: 169, and profit **1735**

  Dynamic optimal solution: `[0, 1, 0, 1, 0, 0, 1]`

  optimal weight: 169, and profit **1735**

  Genetic optimal solution: `[0, 1, 0, 1, 0, 0, 1]`

  optimal weight: 169, and profit **1735**

Knapsack 07

- capacity: *750*
- optimal solution: `[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`
- optimal weight: *749*, and profit: *1458*
  BruteForce optimal solution: `[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`

  optimal weight: 749, and profit **1458**

  Greedy optimal solution: `[1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`

  optimal weight: 740, and profit **1441**

  Branch-And-Bound optimal solution: `[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`

  optimal weight: 749, and profit **1458**

Dynamic optimal solution: `[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`

`optimal weight: 749, and profit **1458**`

Genetic optimal solution: `[1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1]`

`optimal weight: 749, and profit **1458**`

Low-dimension knapsnacks

Knapsack f10_l-d_kp_20_879

- capacity: *879*
- optimal profit: *1025*
  Greedy optimal solution: `[1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]`
  Greedy `exec time 0.00s`
  optimal weight: 837, and profit **1019**
  Branch-And-Bound optimal solution: `[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0]`
  Branch-And-Bound `exec time 0.06s`
  optimal weight: 871.0, and profit **1025.0**
  Dynamic optimal solution: `[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1]`
  Dynamic `exec time 0.09s`
  optimal weight: 871, and profit **1025**
  Genetic optimal solution: `[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1]`
  Genetic `exec time 0.80s`
  optimal weight: 871, and profit **1025**

Knapsack f1_l-d_kp_10_269

- capacity: *269*
- optimal profit: *295*
  Greedy optimal solution: `[0, 1, 1, 0, 1, 0, 0, 1, 1, 1]`
  Greedy `exec time 0.00s`
  optimal weight: 260, and profit **294**
  Branch-And-Bound optimal solution: `[0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0]`
  Branch-And-Bound `exec time 0.05s`
  optimal weight: 269.0, and profit **295.0**
  Dynamic optimal solution: `[0, 1, 1, 1, 0, 0, 0, 1, 1, 1]`
  Dynamic `exec time 0.02s`
  optimal weight: 269, and profit **295**
  Genetic optimal solution: `[0, 1, 1, 1, 0, 0, 0, 1, 1, 1]`
  Genetic `exec time 0.04s`
  optimal weight: 269, and profit **295**

Knapsack f2_l-d_kp_20_878

- capacity: *878*
- optimal profit: *1024*
  Greedy optimal solution: `[1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1]`
  Greedy `exec time 0.00s`
  optimal weight: 837, and profit **1018**
  Branch-And-Bound optimal solution: `[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0]`
  Branch-And-Bound `exec time 0.15s`
  optimal weight: 871.0, and profit **1024.0**
  Dynamic optimal solution: `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1]`
  Dynamic `exec time 0.19s`
  optimal weight: 871, and profit **1024**
  Genetic optimal solution: `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1]`
  Genetic `exec time 1.18s`
  optimal weight: 871, and profit **1024**

Knapsack f3_l-d_kp_4_20

- capacity: *20*
- optimal profit: *35*
  Greedy optimal solution: `[1, 1, 0, 1]`
  Greedy `exec time 0.00s`
  optimal weight: 18, and profit **35**
  Branch-And-Bound optimal solution: `[1.0, 1.0, 0.0, 1.0]`
  Branch-And-Bound `exec time 0.00s`
  optimal weight: 18.0, and profit **35.0**
  Dynamic optimal solution: `[1, 1, 0, 1]`
  Dynamic `exec time 0.00s`
  optimal weight: 18, and profit **35**
  Genetic optimal solution: `[1, 1, 0, 1]`
  Genetic `exec time 0.00s`
  optimal weight: 18, and profit **35**

Knapsack f4_l-d_kp_4_11

- capacity: *11*
- optimal profit: *23*
  Greedy optimal solution: `[1, 1, 0, 0]`
  Greedy `exec time 0.00s`
  optimal weight: 6, and profit **16**
  Branch-And-Bound optimal solution: `[0.0, 1.0, 0.0, 1.0]`
  Branch-And-Bound `exec time 0.01s`
  optimal weight: 11.0, and profit **23.0**
  Dynamic optimal solution: `[0, 1, 0, 1]`
  Dynamic `exec time 0.00s`
  optimal weight: 11, and profit **23**
  Genetic optimal solution: `[0, 1, 1, 0]`

Genetic `exec time 0.00s`
optimal weight: 10, and profit **22**

Knapsack f6_l-d_kp_10_60

- capacity: *60*
- optimal profit: *52*
  Greedy optimal solution: `[0, 0, 1, 0, 1, 1, 1, 1, 1, 1]`
  Greedy `exec time 0.00s`
  optimal weight: 57, and profit **52**
  Branch-And-Bound optimal solution: `[0.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]`
  Branch-And-Bound `exec time 0.02s`
  optimal weight: 57.0, and profit **52.0**
  Dynamic optimal solution: `[0, 0, 1, 1, 1, 0, 1, 0, 0, 0]`
  Dynamic `exec time 0.00s`
  optimal weight: 60, and profit **52**
  Genetic optimal solution: `[0, 0, 1, 0, 1, 1, 1, 1, 1, 1]`
  Genetic `exec time 0.00s`
  optimal weight: 57, and profit **52**

Knapsack f7_l-d_kp_7_50

- capacity: *50*
- optimal profit: *107*
  Greedy optimal solution: `[1, 1, 0, 0, 1, 1, 0]`
  Greedy `exec time 0.00s`
  optimal weight: 48, and profit **102**
  Branch-And-Bound optimal solution: `[1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0]`
  Branch-And-Bound `exec time 0.01s`
  optimal weight: 50.0, and profit **107.0**
  Dynamic optimal solution: `[1, 0, 0, 1, 0, 0, 0]`
  Dynamic `exec time 0.00s`
  optimal weight: 50, and profit **107**
  Genetic optimal solution: `[1, 0, 0, 1, 0, 0, 0]`
  Genetic `exec time 0.00s`
  optimal weight: 50, and profit **107**

Knapsack f8_l-d_kp_23_10000

- capacity: *10000*
- optimal profit: *9767*
  Greedy optimal solution: `[1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0]`
  Greedy `exec time 0.00s`
  optimal weight: 9750, and profit **9751**
  Branch-And-Bound
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 228.01s`
  Dynamic optimal solution: `[1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]`

Dynamic `exec time 0.56s`
optimal weight: 9768, and profit **9767**
Genetic optimal solution: `[1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0]`
Genetic `exec time 0.74s`
optimal weight: 9765, and profit **9764**

Knapsack f9_l-d_kp_5_80

- capacity: *80*
- optimal profit: *130*
  Greedy optimal solution: `[1, 1, 1, 1, 0]`
  Greedy `exec time 0.00s`
  optimal weight: 60, and profit **130**
  Branch-And-Bound optimal solution: `[1.0, 1.0, 1.0, 1.0, 0.0]`
  Branch-And-Bound `exec time 0.00s`
  optimal weight: 60.0, and profit **130.0**
  Dynamic optimal solution: `[1, 1, 1, 1, 0]`
  Dynamic `exec time 0.00s`
  optimal weight: 60, and profit **130**
  Genetic optimal solution: `[1, 0, 1, 1, 1]`
  Genetic `exec time 0.00s`
  optimal weight: 71, and profit **118**

Large-scale knapsnacks

Knapsack knapPI_1_1000_1000_1

- capacity: *5002*
- optimal weight: *5002*, optimal profit: *54503*
  Greedy `exec time 0.00s`
  optimal weight: 4991, and profit **54386**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 229.80s`
  Dynamic `exec time 14.89s`
  optimal weight: 5002, and profit **54503**
  Genetic `exec time 37.74s`
  optimal weight: 4431, and profit **9319**

Knapsack knapPI_1_100_1000_1

- capacity: *995*
- optimal weight: *985*, optimal profit: *9147*
  Greedy `exec time 0.00s`
  optimal weight: 908, and profit **8817**
  Branch-And-Bound `exec time 2.71s`
  optimal weight: 985.0, and profit **9147.0**

Dynamic `exec time 0.19s`
optimal weight: 985, and profit **9147**
Genetic `exec time 3.43s`
optimal weight: 984, and profit **3892**

Knapsack knapPI_1_2000_1000_1

- capacity: *10011*
- optimal weight: *10011*, optimal profit: *110625*
  Greedy `exec time 0.00s`
  optimal weight: 9996, and profit **110547**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 472.64s`
  Dynamic `exec time 61.41s`
  optimal weight: 10011, and profit **110625**
  Genetic `exec time 70.20s`
  optimal weight: 9443, and profit **14607**

Knapsack knapPI_1_200_1000_1

- capacity: *1008*
- optimal weight: *987*, optimal profit: *11238*
  Greedy `exec time 0.02s`
  optimal weight: 981, and profit **11227**
  Branch-And-Bound `exec time 10.17s`
  optimal weight: 987.0, and profit **11238.0**
  Dynamic `exec time 0.40s`
  optimal weight: 987, and profit **11238**
  Genetic `exec time 6.79s`
  optimal weight: 960, and profit **4239**

Knapsack knapPI_1_5000_1000_1

- capacity: *25016*
- optimal weight: *25016*, optimal profit: *276457*
  Greedy `exec time 0.01s`
  optimal weight: 25008, and profit **276379**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 548.42s`
  Dynamic `exec time 407.23s`
  optimal weight: 25016, and profit **276457**
  Genetic `exec time 185.94s`
  optimal weight: 24796, and profit **24169**

Knapsack knapPI_1_500_1000_1

- capacity: *2543*
- optimal weight: *2543*, optimal profit: *28857*
  Greedy `exec time 0.02s`
  optimal weight: 2528, and profit **28834**

Branch-And-Bound `exec time 25.33s`
optimal weight: 2543.0, and profit **28857.0**
Dynamic `exec time 3.11s`
optimal weight: 2543, and profit **28857**
Genetic `exec time 18.08s`
optimal weight: 2541, and profit **7810**

Knapsack knapPI_2_1000_1000_1

- capacity: *5002*
- optimal weight: *5002*, optimal profit: *9052*
  Greedy `exec time 0.00s`
  optimal weight: 4994, and profit **9046**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 265.41s`
  Dynamic `exec time 11.92s`
  optimal weight: 5002, and profit **9052**
  Genetic `exec time 35.06s`
  optimal weight: 4997, and profit **5481**

Knapsack knapPI_2_100_1000_1

- capacity: *995*
- optimal weight: *991*, optimal profit: *1514*
  Greedy `exec time 0.00s`
  optimal weight: 983, and profit **1487**
  Branch-And-Bound `exec time 8.25s`
  optimal weight: 991.0, and profit **1514.0**
  Dynamic `exec time 0.15s`
  optimal weight: 991, and profit **1514**
  Genetic `exec time 3.73s`
  optimal weight: 951, and profit **1265**

Knapsack knapPI_2_2000_1000_1

- capacity: *10011*
- optimal weight: *10010*, optimal profit: *18051*
  Greedy `exec time 0.00s`
  optimal weight: 10010, and profit **18038**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 419.24s`
  Dynamic `exec time 47.54s`
  optimal weight: 10010, and profit **18051**
  Genetic `exec time 80.08s`
  optimal weight: 9889, and profit **10288**

Knapsack knapPI_2_200_1000_1

- capacity: *1008*

- optimal weight: *1006*, optimal profit: *1634*
  Greedy `exec time 0.03s`
  optimal weight: 1004, and profit **1604**
  Branch-And-Bound `exec time 52.73s`
  optimal weight: 1006.0, and profit **1634.0**
  Dynamic `exec time 0.39s`
  optimal weight: 1006, and profit **1634**
  Genetic `exec time 7.46s`
  optimal weight: 987, and profit **1270**

Knapsack knapPI_2_5000_1000_1

- capacity: *25016*
- optimal weight: *25016*, optimal profit: *44356*
  Greedy `exec time 0.01s`
  optimal weight: 25016, and profit **44351**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 482.38s`
  Dynamic `exec time 311.41s`
  optimal weight: 25016, and profit **44356**
  Genetic `exec time 193.92s`
  optimal weight: 24042, and profit **24708**

Knapsack knapPI_2_500_1000_1

- capacity: *2543*
- optimal weight: *2543*, optimal profit: *4566*
  Greedy `exec time 0.13s`
  optimal weight: 2538, and profit **4552**
  Branch-And-Bound `exec time 105.44s`
  optimal weight: 2543.0, and profit **4566.0**
  Dynamic `exec time 2.74s`
  optimal weight: 2543, and profit **4566**
  Genetic `exec time 20.31s`
  optimal weight: 2521, and profit **2874**

Knapsack knapPI_3_1000_1000_1

- capacity: *4990*
- optimal weight: *4990*, optimal profit: *14390*
  Greedy `exec time 0.00s`
  optimal weight: 4974, and profit **14374**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 229.42s`
  Dynamic `exec time 10.98s`
  optimal weight: 4990, and profit **14390**
  Genetic `exec time 42.87s`
  optimal weight: 4963, and profit **6863**

Knapsack knapPI_3_100_1000_1

- capacity: *997*
- optimal weight: *997*, optimal profit: *2397*
  Greedy `exec time 0.00s`
  optimal weight: 975, and profit **2375**
  Branch-And-Bound `exec time 5.85s`
  optimal weight: 997.0, and profit **2397.0**
  Dynamic `exec time 0.19s`
  optimal weight: 997, and profit **2397**<br **Genetic** `exec time 4.22s`
  optimal weight: 970, and profit **1470**

Knapsack knapPI_3_2000_1000_1

- capacity: *9819*
- optimal weight: *9819*, optimal profit: *28919*
  Greedy `exec time 0.00s`
  optimal weight: 9727, and profit **28827**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 423.94s`
  Dynamic `exec time 47.75s`
  optimal weight: 9819, and profit **28919**
  Genetic `exec time 84.89s`
  optimal weight: 9650, and profit **12150**

Knapsack knapPI_3_200_1000_1

- capacity: *997*
- optimal weight: *997*, optimal profit: *2697*
  Greedy `exec time 0.02s`
  optimal weight: 949, and profit **2649**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 228.41s`
  Dynamic `exec time 0.33s`
  optimal weight: 997, and profit **2697**
  Genetic `exec time 8.55s`
  optimal weight: 993, and profit **1493**

Knapsack knapPI_3_5000_1000_1

- capacity: *24805*
- optimal weight: *24805*, optimal profit: *72505*
  Greedy `exec time 0.01s`
  optimal weight: 24746, and profit **72446**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 478.12s`
  Dynamic `exec time 289.82s`
  optimal weight: 24805, and profit **72505**
  Genetic `exec time 219.38s`
  optimal weight: 23471, and profit **29671**

Knapsack knapPI_3_500_1000_1

- capacity: *2517*
- optimal weight: *2517*, optimal profit: *7117*
  Greedy `exec time 0.02s`
  optimal weight: 2498, and profit **7098**
  Branch-And-Bound
  Branch-And-Bound skipped after `exec time 228.64s`
  Dynamic `exec time 2.67s`
  optimal weight: 2517, and profit **7117**
  Genetic `exec time 17.19s`
  optimal weight: 2495, and profit **3495**

Knapsack knapPI_1_10000_1000_1

- capacity: *49877*
- optimal weight: *49877*, optimal profit: *563647*
  Greedy `exec time 0.02s`
  optimal weight: 49876, and profit **563605**
  Branch-And-Bound `exec time 1.34s`
  optimal weight: 49877.0, and profit **563641.0**
  Genetic `exec time 529.28s`
  optimal weight: 23150, and profit **27414**

Knapsack knapPI_2_10000_1000_1

- capacity: *49877*
- optimal weight: *49877*, optimal profit: *90204*
  Greedy `exec time 0.02s`
  optimal weight: 49877, and profit **90200**
  Branch-And-Bound `exec time 0.98s`
  optimal weight: 49874.0, and profit **90198.0**
  Genetic `exec time 730.94s`
  optimal weight: 30665, and profit **31779**

Knapsack knapPI_3_10000_1000_1

- capacity: *49519*
- optimal weight: *49519*, optimal profit: *146919*
  Greedy `exec time 0.04s`
  optimal weight: 49488, and profit **146888**
  Branch-And-Bound `exec time 0.85s`
  optimal weight: 49519.0, and profit **146919.0**
  Genetic `exec time 526.79s`
  optimal weight: 20570, and profit **25070**

Comparison:

benchmark	algorithm	execution mean	execution std	capacity	optim_weight	optim_profit
1	Branch-And-Bound	0	0	165	165	309
1	BruteForce	0.0016	0.0005	165	165	309
1	Dynamic	0.006	0.0007	165	165	309
1	Genetic	0.0145	0.0089	165	165	309
1	Greedy	0	0	165	165	309
2	Branch-And-Bound	0	0	26	26	51
2	BruteForce	0	0	26	26	51
2	Dynamic	0.0006	0.0005	26	26	51
2	Genetic	0.0004	0.0005	26	24	47
2	Greedy	0	0	26	23	47
3	Branch-And-Bound	0.0002	0.0005	190	190	150
3	BruteForce	0	0	190	190	150
3	Dynamic	0.0044	0.0006	190	190	150
3	Genetic	0.0004	0.0005	190	172	119
3	Greedy	0	0	190	179	146
4	Branch-And-Bound	0	0	50	50	107
4	BruteForce	0.0006	0.0005	50	50	107
4	Dynamic	0.0012	0.0004	50	50	107
4	Genetic	0.0014	0.0009	50	50	107
4	Greedy	0	0	50	48	102
5	Branch-And-Bound	0	0	104	104	900
5	BruteForce	0.0004	0.0005	104	104	900
5	Dynamic	0.0038	0.0008	104	104	900
5	Genetic	0.0032	0.0008	104	103	898
5	Greedy	0	0	104	97	858
6	Branch-And-Bound	0.0002	0.0004	170	169	1735
6	BruteForce	0.0002	0.0005	170	169	1735
6	Dynamic	0.0052	0.0008	170	169	1735
6	Genetic	0.001	0.0007	170	169	1735
6	Greedy	0	0	170	140	1478
7	Branch-And-Bound	0.0042	0.0008	750	749	1458
7	BruteForce	0.0541	0.0147	750	749	1458
7	Dynamic	0.058	0.0047	750	749	1458
7	Genetic	0.306	0.0365	750	749	1458
7	Greedy	0.0002	0.0004	750	740	1441