Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Engineering
and Networks Laboratory

Bio-inspired Computation and Optimization

Project 1: Knapsack Problem – Part I

- Johannes Bader
- Office ETZ G 81
- Phd student since November 2006
- baderj@tik.ee.ethz.ch

http://www.tik.ee.ethz.ch/sop/education/lectures/BOD/

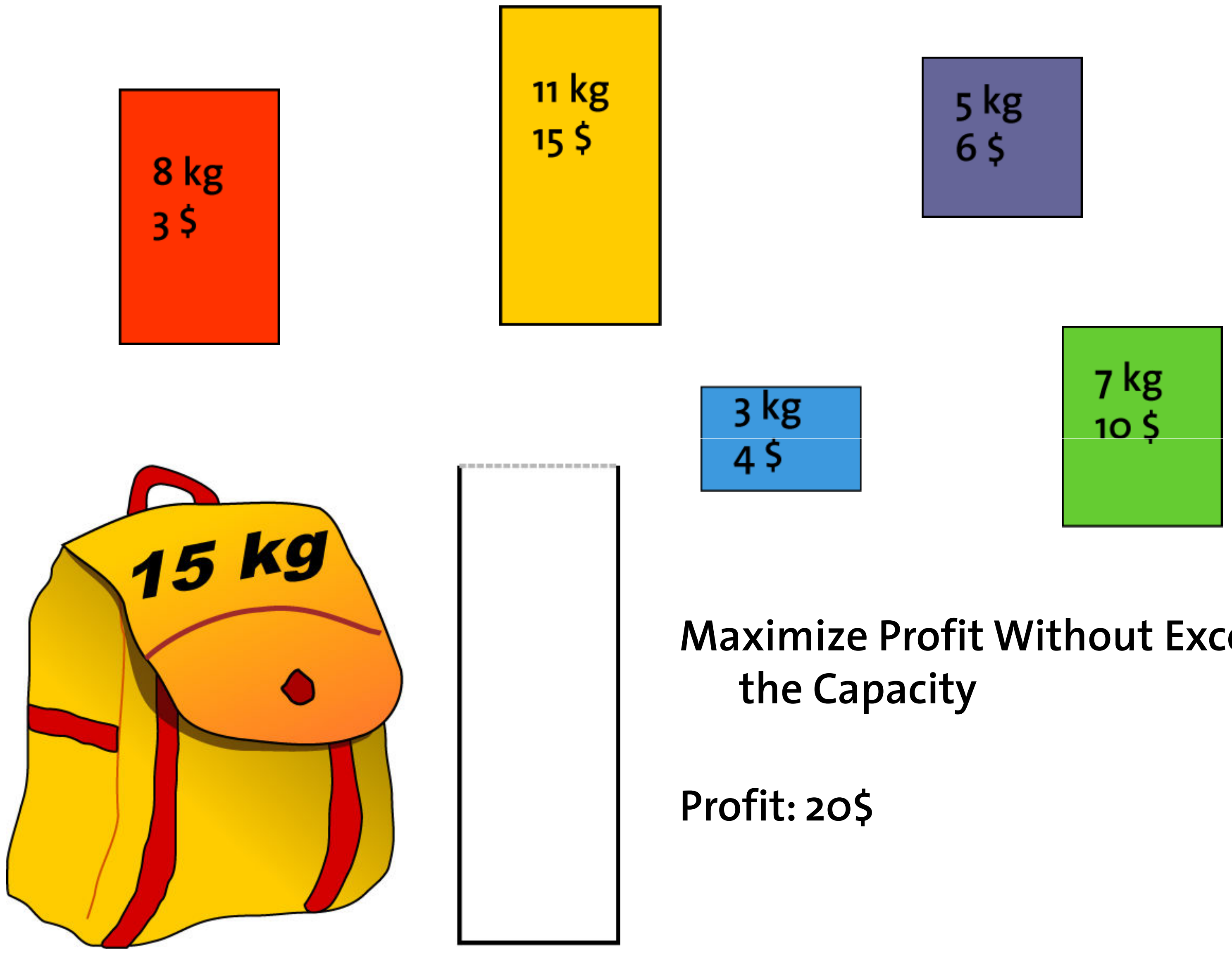
Exercises

- Class Time: 15.15 – 16.00
- Four Projects:
  1. supervised by me (Johannes)
  2. supervised by Tamara Ulrich
  3. supervised by Mattia Gazzola
  4. supervised by Mattia Gazzola
- Discussion of the exercises
- Time to ask questions also concerning the lecture!
  - during the exercises
  - after the exercise
  - by e-mail
  - arrange an appointment

General Guidelines

- The projects can be worked on in groups of maximal three students.
- In order to fulfill the “Testat” conditions, at least 40 points need to be reached for each project and 200 points in total for all 4 projects. Tasks which can easily be skipped are marked as supplementary tasks.
- The solutions have to be submitted by
  - Task 1: March 5th at 24:00
  - Task 2: March 12th at 24:00
- The solutions (program code, plots, text, ...) have to be submitted via email. As attachments only PDF, Word or text documents are accepted. Only submit one single document.
- The solutions will be graded and returned in the following week’s exercise lesson.

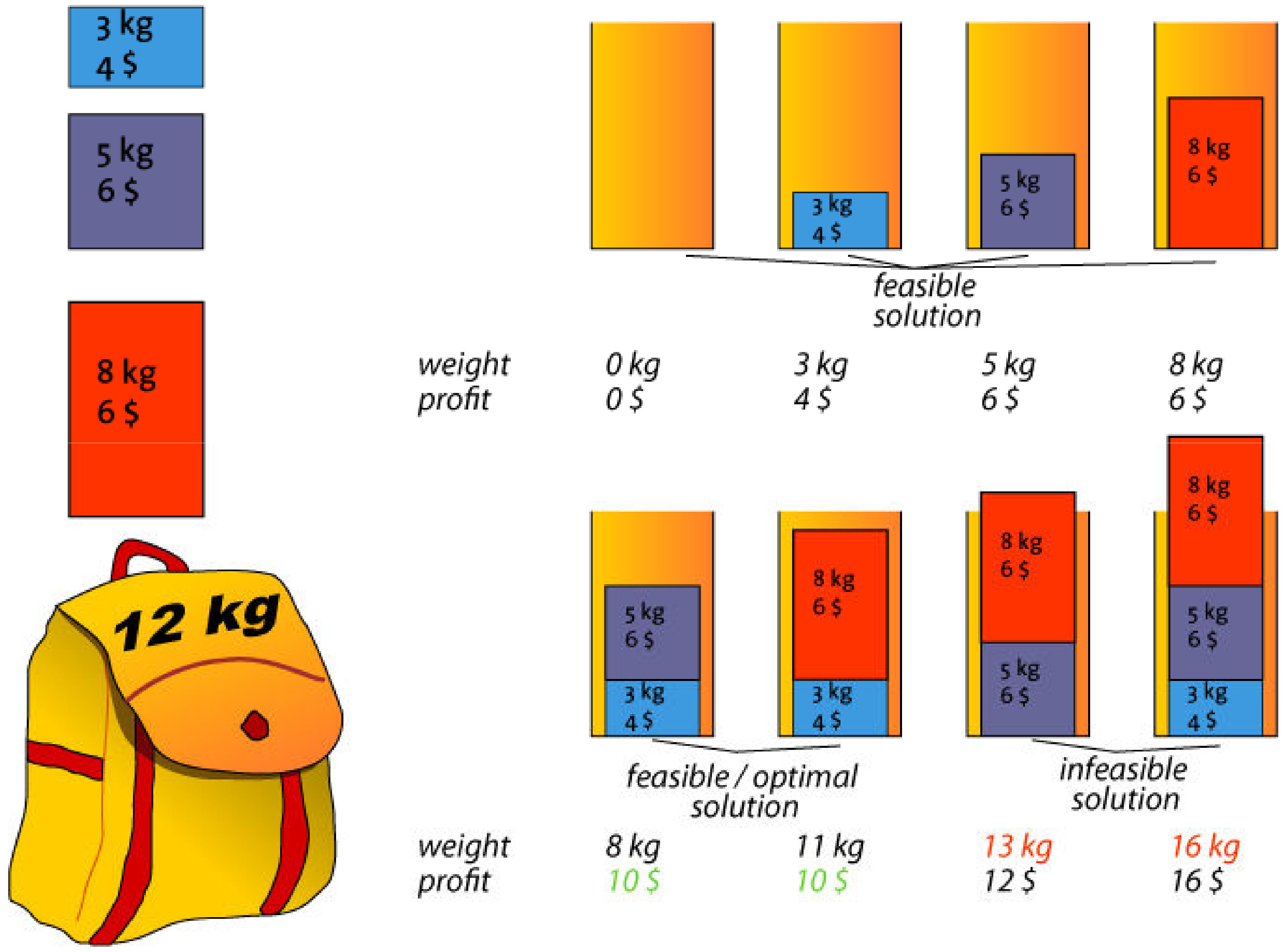
Knapsack Problem



Maximize Profit Without Exceeding the Capacity

Profit: 20$

Feasible, Infeasible and Optimal Solutions



3 kg
4 $
5 kg
6 $
8 kg
6 $
12 kg
feasible solution
weight 0 kg 3 kg 5 kg 8 kg
profit 0 $ 4 $ 6 $ 6 $
feasible / optimal solution
infeasible solution
weight 8 kg 11 kg 13 kg 16 kg
profit 10 $ 10 $ 12 $ 16 $

Task 1: Modelling and Implementation

- Describe how you would represent a solution *x* for the knapsack problem and define the search space *X*.
- Give an expression for the objective function *f*. Solutions that exceed the given capacity bound shall receive an objective function value of zero.
- Describe an algorithm to calculate the objective function value of a given solution *x* in pseudocode.
- Choose a programming language. Write a program that is able to read all problem parameters (number of items, profit and weight values, capacity) from text files.
- Generate *10000* random solutions and calculate the objective function value of each of them.
- **Supplementary task:** Extend your program such that it enumerates all possible solutions and prints out the optimal ones. For each problem instance from task d) report the optimal solution(s), their objective function value and the running time..

Task 2: Neighborhood Search

a) Define a parameterized neighborhood function $N_d(x), x \in X, d \in \mathbf{N}$, for the knapsack problem. The parameter d should reflect the size of the neighborhood. A neighborhood function should have the property that for any $x \in X, d < n : N_d(x) \subset N_{d'}(x) \Leftrightarrow d < d'$. Show that this property holds for your function. *(10 Points)*

b) Implement the Randomized Local Search for the knapsack problem. Run the program on the problem instance given on the lecture website for 100000 iterations and 5 different d values. Plot the current best function value versus the elapsed number of iterations for each of the five runs in the same diagram and store the final best solution found x^*. *(15 Points)*

Example: Random Search

```
1:  Randomly choose an initial solution $x_1$ from $X$
2:  Calculate $f(x_1)$
3:  Initialize memory, i.e., $M = \{(x_1, f(x_1))\}$
4:  Set iteration counter $t = 0$
5:  loop
6:      $t = t + 1$
7:      Randomly choose $x_t \in X$
8:      Calculate $f(x_t)$
9:      if $f(x_t) \geq f(x)$ where $M = \{(x, f(x))\}$ then
10:         Keep $x_t$ as best solution seen so far, i.e., $M = \{(x_t, f(x_t))\}$
11:     end if
12:     if $t \geq t_{MAX}$ then
13:         Output solution $x$ stored in $M$ with $M = \{(x, f(x))\}$
14:         Stop
15:     end if
16: end loop
```

Maximization problem:

$(X, \Re, f, \geq)$

Random search does not make use of previously visited solutions.

Task 2 cont.

c) **Supplementary task:** We want to measure how the distance from the best solution x^* relates to the difference in objective function value. Create 10000 random solutions from $N_d(x^*)$ for a reasonable maximal distance d. Plot the solutions in a diagram, where the x-axis denotes the distance in objective space $f(x^*) - f(x)$ and the y-axis the distance in the search space according to the chosen neighborhood function, i.e. $min\{d : x \in N_d(x^*)\}$. Describe the plot and discuss what it reveals about the structure of the problem and the applicability of neighborhood search. *(15 Points)*

Task 2: Neighborhood Search

d) **Supplementary task:** Contrarily to the Randomized Local Search, the Metropolis algorithm accepts a worse solution with a certain fixed probability T. Implement the Metropolis algorithm for the knapsack problem using the neighborhood function from task 2a). Run the program on the problem instance given on the lecture website for 5 different T values and 100000 iterations each. Assume $d = 1$. Plot the current best objective value versus the elapsed number of iterations for all 5 runs in the same diagram. Also plot the current objective value versus the number of iterations. What T value do you recommend and why? *(20 Points)*

$$\textbf{if } f(x_t) \geq f(x) \ \vee \ rand[0,1] \leq e^{-\frac{f(x)-f(x_t)}{T}} \ \textbf{then}$$
$$M = \{(x_t, f(x_t))\}$$