Prof. Friedrich Eisenbrand
Location: MA A3 31

Question session: 24.11.10
Discussion: 1.12.10

Exercises

Optimization Methods in Finance

Fall 2010

Sheet 5

Note: This is just one way, a solution could look like. We do not guarantee correctness. It is your task to find and report mistakes.

Exercise 5.1 (*)

We consider European call options $i = 1, \ldots, n$ with respect to the same underlying security and with the same maturity. The ith option has strike price K_i. In the lecture, so far we considered the case that the price to buy or sell an option is the same. However, in reality there is the socalled *bid/ask spread*. That means at time 0 we can buy an option for S_a^i (ask price) and we can sell it for S_b^i (bid price), but $S_a^i > S_b^i$ (still the value of the option i at time 1 will be $\max\{S_1 - K_i, 0\}$). Can you design a linear program, which detects type A arbitrage in this setting?

Solution:

Let x_i^+ be the number times that we buy option i for our portfolio and let x_i^- the number of times that we sell option i. The LP to detect type A arbitrage is

$$\begin{aligned}
\min \sum_{i=1}^{n} (S_a^i x_i^+ - S_b^i x_i^-) & \\
\sum_{i=1}^{n} x_i \max\{S_1 - K_i, 0\} &\geq 0 \quad \forall S_1 \in \{K_1, \ldots, K_n\} \\
\sum_{i=1}^{n} x_i &\geq 0 \\
x &= x_i^+ - x_i^- \quad \forall i = 1, \ldots, n \\
x_i^+, x_i^- &\geq 0 \quad \forall i = 1, \ldots, n
\end{aligned}$$

Again, if there is a portfolio x with objective funtion < 0, then it provides arbitrage.

Exercise 5.2 (*)

Suppose we are given a map of canton Vaud with n many villages. For villages i and j, let $c_{ij} \geq 0$ be the length of the road from i to j (or $c_{ij} = \infty$ if no direct road exists). Consider table entries

$A(i, j, k) =$ length of the shortest path from i to j, where we use at most k many intermediate stations

Give Bellman equations to compute $A(i, j, k)$. How is the base case $A(i, j, 0)$ defined? How can we read the length of the shortest route (using arbitrarily many interstations) from i to j from the table?

Solution:

The bases case is $A(i,j,0) = c_{ij}$ (might be ∞). Then

$$A(i,j,k) = \min_{\ell=1,\ldots,n} \{c_{i\ell} + A(\ell,j,k-1)\}$$

(we assume that $c_{ii} = 0$ for all i). The shortest route from i to j will never visit a village twice, since $c_{ij} \geq 0$. Hence $A(i,j,n)$ gives the length of the best tour (in fact, even $A(i,j,n-2)$ would work).

Exercise 5.3 (*)

Suppose we have a warehouse to store $\{0,\ldots,B\}$ units of a good. At time 1 the warehouse is empty. In period $i = 1,\ldots,n$ (which lies between time i and time $i+1$) our company consumes $w(i) \in \mathbb{N}_0$ many units of the good. At any time $i = 1,\ldots,n$ we can order an arbitrary integer amount between 0 and $B \in \mathbb{N}$ of the good (just the capacity of the warehouse may not be exceeded). For simplicity we assume that the ordered goods arrive immediately. The cost that we have to pay for the ordered goods possibly reflects *economies of scale*, that means ordering a larger amount might decrease the price per unit. On the other hand prices change over time (but are known in advance). Thus let $c(i,k) \in \mathbb{N}_0$ be the amount, that we pay for $k \in \mathbb{N}_0$ units at time i (you may assume that $c(i,k) \leq c(i,k+1)$). Our aim is to order goods such that the cumulated cost is minimized and we can fulfill the demands in all time periods.

1. Introduce suitable table entries (suitable to compute the optimum solution in polynomial time in $n, B, c(i,k)$ via dynamic programming)
   Hint: You will need a 2-dim table, like $v(i,j)$

2. State the Bellman equations, needed to compute the table entries.

3. Compute the table entries for $n = 3, B = 2, w = (1,0,2)$. What is the optimum solution and its value if the costs are

$c(i,k)$	$k=0$	$k=1$	$k=2$
$i=1$	0	2	3
$i=2$	0	1	2
$i=3$	0	2	3

Solution:

For (1). We use

$$\begin{aligned} v(i,j) &= \text{minimal costs to have the warehouse filled with exactly } j \\ &\quad \text{units at time } i \text{ and having all demands} \\ &\quad \text{in prior periods satisfied} \end{aligned}$$

For (2). The Bellman equations are

$$\begin{aligned} v(1,j) &= c(1,j) \quad \forall j = 0,\ldots,B \\ v(i,j) &= \infty \quad \forall i = 2,\ldots,n, \quad \forall j < 0 \\ v(i,j) &= \min_{k=0,\ldots,j} \{v(i-1, j-k+w(i-1)) + c(i,k)\} \quad \forall i = 2,\ldots,n \quad \forall j = 0,\ldots,B \end{aligned}$$

For (3).

$v(i,j)$	$j=0$	$j=1$	$j=2$
$i=1$	0	2	3
$i=2$	2	3	4
$i=3$	2	3	4
$i=4$	**4**		

using

$$\begin{aligned}
v(2,0) &= \min\{v(1,1)+c(2,0)\} = 2+0 = 2\\
v(2,1) &= \min\{v(1,2)+c(2,0), v(1,1)+c(2,1)\} = \min\{3+0, 2+1\} = 3\\
v(2,2) &= \min\{v(1,2)+c(2,1), v(1,1)+c(2,2)\} = \min\{3+1, 2+2\} = 4\\
v(3,0) &= \min\{v(2,0)+c(3,0)\} = 2\\
v(3,1) &= \min\{v(2,1)+c(3,0), v(2,0)+c(3,1)\} = \min\{3+0, 2+2\} = 3\\
v(3,2) &= \min\{v(2,2)+c(3,0), v(2,1)+c(3,1), v(2,0)+c(3,2)\} = \min\{4+0, 3+2, 2+3\} = 4
\end{aligned}$$

The optimum cost is $v(3,2) = 4$ (since we need $w(3) = 2$ units in period 4). The optimum solution is $x = (2,1,0)$ or $x = (1,2,0)$ (buy x_i units at time i).

Exercise 5.4 (*)

Suppose our company has a budget of $B \in \mathbb{N}$ and we have the possibility to invest into projects in several countries. Suppose that the number of projects in country $j \in \{1,\ldots,C\}$ is n_j. The ith project in country j needs an investment of $a_{ij} \in \{0,\ldots,B\}$ (at time 0) and will give a profit of $p_{ij} \in \{0,\ldots,P\}$ at time 1. All projects are $0/1$-decisions, meaning we can take it only once (or leave it). For the sake of simplicity we assume that the part of the budget that is not invested is lost. Unfortunately, each country raises taxes. If we make a total gross profit of p in country j, we will receive a net profit of just $t_j(p) \in \{0,\ldots,p\}$ (the difference $p - t_j(p)$ are taxes). Here $t_j : \mathbb{Z}_+ \to \mathbb{Z}_+$ is a non-decreasing function that depends on the country.

Our goal is to select a subset of the project, such that the total investment does not exceed B and we maximize our global profit *after taxes*. Show how you can solve this problem with dynamic programming.

Solution:

We aim to solve

$$\max\Big\{ \underbrace{\sum_{j=1}^{C} f_j(p_j)}_{\text{net profit from country } j} \mid I_j \subseteq \{1,\ldots,k_C\}\ \forall j = 1,\ldots C; \sum_{j=1}^{C}\sum_{i\in I_j} a_{ij} \le B;\ p_j := \sum_{i\in I_j} p_{ij} \Big\}$$

We create table entries

$$\begin{aligned}
A_j(i,B') &= \max\Big\{ \sum_{i'\in I} p_{i'j} \mid I \subseteq \{1,\ldots,i\} : \sum_{i'\in I} a_{ij} \le B' \Big\}\\
&= \text{max. gross profit from investing } \le B' \text{ into projects } 1,\ldots,i \text{ of country } j
\end{aligned}$$

Then

$$\begin{aligned}
A_j(0,B') &= 0\\
A_j(i,B') &= \max\big\{A_j(i-1,B'), A_j(i-1,B'-a_{ij}) + p_{ij}\big\}
\end{aligned}$$

Now that we have $A_j(i,B')$, we consider a second type of table entries

$$\begin{aligned} D(i,B') &= \max\Big\{\sum_{j=1}^{C} A_j(n_j,B_j) \mid B_1+\ldots+B_i \le B\Big\} \\ &= \text{maximum net profit from investing } \le B' \text{ into projects in countries } 1,\ldots,j \end{aligned}$$

Those entries can be computed via

$$\begin{aligned} D(0,B') &= 0 \\ D(j,B') &= \max_{B''\in\{0,\ldots,B'\}} \{A_j(k_c,B'') + D(j-1,B'-B'')\} \end{aligned}$$

Overall $D(C,B)$ will give the final answer.

Exercise 5.5 (Practical exercise - 1 point)
Our friend *Bruno the Burglar* plans to brake into the main safe of the UB Credit Bank. From a reliable informant he knows that the safe contains n items, where the informant also told the exact weight $w_i \in \mathbb{N}$ (say in 100gr) and value $p_i \in \mathbb{N}$ (say in thousand CHF) for each $i = 1,\ldots,n$. Unfortunately Bruno can carry only B weight units in his bag. Since we want to help our friend, you have to *implement a dynamic programming algorithm* such that the cumulated value of the taken items is maximized, while the weight bound of B is not exceeded. In other words, solve

$$\max\Big\{\sum_{i=1}^{n} x_i p_i \mid \sum_{i=1}^{n} x_i w_i \le B;\ x_i \in \{0,1\} \quad \forall i = 1,\ldots,n\Big\}$$

by dynamic programming. In C syntax the instance is

```
int n = 40;
long B = 5000;
int p[] = {1477, 1175,  735, 2270,  548, 2300, 2018,  431,  422,  410,
           2593,  805, 1036, 2043,  664, 1753, 2511,  961, 2044, 1261,
           3451, 1849,  387,  775,  259, 2673, 1600, 2163, 1072,  684,
           2087,  888, 1686, 1142, 1478, 2252, 1051,  488, 3796, 1091};
int w[] = { 323,  333,  234,  447,  246,  298,  285,  225,  311,  409,
            349,  476,  441,  268,  241,  228,  459,  408,  315,  405,
            451,  458,  287,  438,  223,  490,  268,  460,  300,  358,
            353,  375,  339,  272,  290,  456,  463,  464,  481,  403};
```

1. You can implement the algorithm in one of the programming languages C/C++/Java/Pascal/Basic/Matlab (you can choose your favourite one).

2. Your submission should contain

   - A short description which table entries you use and which recursion you use to compute them.
   - Your (compilable) code
   - The value of the found optimum solution and selected items in it.

3. Send the files till **1.12.10 (23:59h)** to thomas.rothvoss@epfl.ch.
4. You can work in groups up to 3 people (you need only one submission per group).