# Econ 172A - Slides from Lecture 11

Joel Sobel

November 6, 2012

## Announcements

- Problems on Branch and Bound: 2007: Homework 3, #1; Final: #5,6 2008: Homework 3 # 1, 2; Final: #4.
- Midterm:
  Graded midterms available in Section on Nov. 6 and in class on Nov. 8.
  Please consult posted answers and regrading instructions.

## Knapsack Formulation

Given a knapsack with fixed capacity and a collection of items, each with a weight and value, find the items to put in the knapsack that maximizes the total value carried subject to the requirement that that weight limitation not be exceed.

- Data:
  1. $N$, (integer) number of items
  2. $C$, the capacity of the knapsack
  3. For each $j = 1, \ldots, N$, the value $v_j > 0$
  4. and weight $w_j > 0$ of item $j$.
- Figure out which items to pack to maximize value carried subject to capacity constraint.

## Trick

Let $x_j$ be a variable that is equal to 1 if you place item $j$ in the knapsack and equal to 0 if you do not.

This is a clever way to define variables.

$x_j$ "counts" whether you take item $j$ or not.

Given this definition, writing constraints and objective function is easy (or at least easier).

## Expressions

If $x = (x_1, \ldots, x_N)$ and all of the $x_j$ take on the value 0 or 1, then

$$\sum_{j=1}^{N} v_j x_j$$

is equal to the value of what you put in the knapsack and

$$\sum_{j=1}^{N} w_j x_j$$

is equal to the weight of what you put in the knapsack.

## Formulation

$$\max \sum_{j=1}^{N} v_j x_j$$

subject to

$$\sum_{j=1}^{N} w_j x_j \leq C,$$

$$0 \leq x_j \leq 1,$$

and $x_j$ integer.
In this formulation, except for the restriction that $x_j$ takes an integer values, the constraints are linear. Notice that by requiring that $x_j$ both be between 0 and 1 and be an integer, we are requiring that $x_j$ be either zero or 1.

## Discussion

- Only finitely many feasible vectors in general integer programming problem.
- In knapsack problem, the only possible answers are subset of $N$ things, so there are only $2^N$ possibilities.
- You can solve the problem (knowing the capacity, values, and weights) by looking at each possible subcollection of items, and selecting the one that has the most value as long as the value in no greater than $C$.
- Conceptually easier than LP (which has an infinite feasible set).
- Problem: The number of things to check grows "exponentially" with $N$.

## “Rules of Thumb”

An approach to the knapsack problem:

- Order the items in decreasing order of value. Take the most valuable thing that fits. Continue.
- Order the items in decreasing order of “efficiency” (ratio of value to weight). Take the most efficient thing that fits. Continue.

These give you a method for packing the knapsack, but they may not lead to the solution to the problem.

## Why?

Give examples where the rules of thumb fail.

## Branch and Bound

- General Method
- Method is not efficient. (It could take a long time.)
- Instead: systematic way to solve IPs.

## Example

$$\begin{array}{rrcrcrcrcl}
\max & 2x_1 & + & 4x_2 & + & 3x_3 & + & x_4 & & & (0) \\
\text{subject to} & 3x_1 & + & x_2 & + & 4x_3 & + & x_4 & \leq & 3 & (1) \\
 & x_1 & - & 3x_2 & + & 2x_3 & + & 3x_4 & \leq & 3 & (2) \\
 & 2x_1 & + & x_2 & + & 3x_3 & - & x_4 & \leq & 6 & (3)
\end{array}$$

with the additional constraint that each variable be 0 or 1.

## Bound

The first step of the procedure is to find an upper bound for the value of the problem.
Standard method: Solve the "relaxed" problem (without integer constraints).

## Relaxation

$$\begin{array}{rrcrcrcrcl}
\text{max} & 2x_1 & + & 4x_2 & + & 3x_3 & + & x_4 & & & (0) \\
\text{subject to} & 3x_1 & + & x_2 & + & 4x_3 & + & x_4 & \leq & 3 & (1) \\
 & x_1 & - & 3x_2 & + & 2x_3 & + & 3x_4 & \leq & 3 & (2) \\
 & 2x_1 & + & x_2 & + & 3x_3 & - & x_4 & \leq & 6 & (3)
\end{array}$$

and $x \geq 0$.

- Same as before, except integer constraints are gone.
- Value of relaxed problem must be greater than or equal to value of original problem. (Why?)
- Value of relaxed problem is equal to value of original problem exactly when related problem has an integer solution.
- Relaxed problem should be easier to solve than original problem.

## Solution to Relaxed Problem

- $x = (x_1, \dots, x_4) = (0, 1, 1/4, 1)$ with value 23/4.
- You can check this (using complementary slackness).
- Finding the solution requires Excel.
- So value of the original problem is is no more than 23/4.
- In fact, since the value of the original problem must be an integer, the value of the original problem can be at most 5.

## What Next?

- The value of the problem cannot be greater than 5.
- If we could figure out a way to make the value equal to 5 we would be done.
- A guess: Set $x_3 = 0$ (instead of $1/4$). Take $x_2 = x_4 = 1$ and $x_1 = 0$ (as before).
- This is feasible for the original integer program and gives value 5.
- Hence it is a solution.

If you are able to make the guess, then you could quit.

Branch and Bound works even if you cannot guess.

## Branch

- Assume that you didn't find a feasible way to attain the upper bound of 5.
- The original problem has four variables.
- It would be easier to solve it if had three variables.
- Form two related subproblems involving three variables.

## Branching

- Observation: When you solve the problem, either $x_1 = 0$ or $x_1 = 1$.
- So if I can solve two subproblems (one with $x_1 = 0$ and the other with $x_1 = 1$), then I can solve the original problem.
- I obtain subproblem I by setting $x_1 = 0$. This leads to

$$\begin{array}{lrcrcrcrl} \text{max} & 4x_2 & + & 3x_3 & + & x_4 & & & (0) \\ \text{subject to} & x_2 & + & 4x_3 & + & x_4 & \leq & 3 & (1) \\ & -\ 3x_2 & + & 2x_3 & + & 3x_4 & \leq & 3 & (2) \\ & x_2 & + & 3x_3 & - & x_4 & \leq & 6 & (3) \end{array}$$

- This problem is the original problem with $x_1 = 0$.

## The other subproblem

Subproblem II, comes from setting $x_1 = 1$.

$$\begin{array}{lrcrcrcrl}
\text{max} & 4x_2 & + & 3x_3 & + & x_4 & & & (0) \\
\text{subject to} & x_2 & + & 4x_3 & + & x_4 & \leq & 0 & (1) \\
- & 3x_2 & + & 2x_3 & + & 3x_4 & \leq & 2 & (2) \\
 & x_2 & + & 3x_3 & - & x_4 & \leq & 4 & (3)
\end{array}$$

This constitutes the "branching" part of the branch and bound method. Now comes to the bounding part again.

## Bounding Again

- Bound Subproblem I.
- Solve Relaxed Problem I.
- It is the same as Original Relaxed Solution because $x = (0, 1, 1/4, 1)$, the solution to original relaxed problem is feasible for relaxed subproblem I.
- Relaxed subproblem II has solution $(1, 0, 0, 0)$. [Use excel or common sense.]
- Since $(1, 0, 0, 0)$ is integer, it is feasible for subproblem II.
- So we have solved Subproblem II. It has value 2

## What we now know

1. Upper bound is 5.
2. Lower bound is 2.
3. We know everything about Subproblem II.
4. Subproblem I is “easier” than original problem (only three variables).
5. Repeat procedure on Subproblem I (branching).

## Branching Again

- Break Subproblem I into two subproblems.
- Subproblem I.I in which $x_2 = 0$ (and $x_1 = 0$).
- Subproblem I.II in which $x_2 = 1$ (and $x_1 = 0$).
- At this point the remaining problems are probably easy enough to solve by observation: $x_3 = 0$ and $x_4 = 1$ for Subproblem I.I. (This means that the possible solution identified by solving subproblem I.I is $(0, 0, 0, 1)$ with value 1.)
- For Subprogram I.II the solution is also $x_3 = 0$ and $x_4 = 1$, but to get to this subproblem we set $x_2 = 1$, so the possible solution identified from this computation is $(0, 1, 0, 1)$ with value 5. (If you do not see how I obtained the values for $x_3$ and $x_4$, then carry out the branching step one more time.)

## Assessment

At this point, we have the following information:

1. The value of the problem is no more than 5.
2. There are three relevant subproblems.
3. The value of Subproblem II is 2.
4. The value of Subproblem I.I is 2.
5. The value of Subproblem I.II is 5.

Consequently, Subproblem I.II really does provide the solution to the original problem.

## Recap

- This exercise illustrate the branch-and-bound technique, but it does not describe all of the complexity that may arise.
- I will describe the technique in general terms.
- Then another example.