Solving Knapsack Problems via Branch-&-Bound

Dennis L. Bricker
Dept. of Industrial Engineering
University of Iowa
Iowa City, Iowa 52242
e-mail: dbricker@icaen.uiowa.edu

author

As an alternative to dynamic programming (DP), a knapsack problem can be solved by the branch-and-bound approach.

Let's use an example to illustrate the branch-and-bound approach to solving knapsack problems:

Randomly Generated Problem (seed 5354416)

Number of items: 6
Capacity of Knapsack: 39
Maximum units of any item to be included is 1 ←

Although in this example at most one of each item is allowed, the approach applies when more than one is allowed.

i	W	V
1	4	6
2	17	10
3	14	12
4	16	11
5	9	9
6	20	12

W = 'weight' of item
V = value of item

This knapsack problem can be formulated as an integer linear programming problem:

Maximize $6X_1 + 10X_2 + 12X_3 + 11X_4 + 9X_5 + 12X_6$
subject to

$$4X_1 + 17X_2 + 14X_3 + 16X_4 + 9X_5 + 20X_6 \le 39$$
$$X_j \in \{0,1\},\ j = 1,2,\ldots 6$$

LP Relaxation

If we replace the constraint $X_j \in \{0,1\}$ with $0 \le X_j \le 1$, that is, we allow fractional values for the variables as well as zero and one, we have the "LP Relaxation" of the problem.

Because the feasible solutions of the LP Relaxation include the feasible solutions of the integer knapsack problem, the optimal value of the LP Relaxation must be at ***at least as large as*** the optimum of the integer problem.

(That is, if we allow fractions of items to be included in the knapsack as well as whole items, we can do at least as well and generally better!)

LP Relaxation

The LP Relaxation is very easy to solve:

- Compute, for each item, the ratio of (value/weight)
- Sort the items according to this ratio, in descending order

item i	Value V	Weight W	Ratio V/W
1	6	4	1.5
5	9	9	1
3	12	14	0.857143
4	11	16	0.6875
6	12	20	0.6
2	10	17	0.588235

- Fill the knapsack with as many whole items as possible beginning at the top of the sorted list

Items 1, 5, and 3 require 27 units of the available 39 units of capacity; this leaves only 12 units, which is not enough for item 4, next on the list.

LP Relaxation

- Fill the remaining space available in the knapsack with a fraction of the next item on the list, namely the ratio of available space to weight of the next item, i.e.,

$$\frac{CAP - \sum_{j=1}^{k} w_{(j)}}{w_{(k+1)}}$$

where k is the number of whole items placed in the knapsack, and $w_{(j)}$ is the j^{th} item on the sorted list.

In the example, after adding the first three items on the list, 12 units of capacity remain, while the next item on the list (item 4) has a weight of 16. Therefore, we can put 75% of item 4 into the knapsack.

LP Relaxation

LP Relaxation
of Knapsack Problem

Randomly Generated Problem (seed 5354416)

i	V	W	V/W	X
1	6	4	1.5	1
5	9	9	1	1
3	12	14	0.857143	1
4	11	16	0.6875	0.75
6	12	20	0.6	0
2	10	17	0.588235	0

Total value of knapsack contents: 35.25
(This is an upper bound on the optimal integer solution)
Rounding down yields value 27, which is a lower bound on the optimum.)

Notice the LOWER BOUND that is readily obtained by rounding down the fractional variable to zero.

We will use both these upper & lower bounds in the "branch-&-bound" algorithm:

- the lower bound & its associated integer solution in order to get "good" solutions to the problem, the best of which will be optimal
- the upper bound in order to eliminate some new "subproblems" which are created by "branching". (Subproblems not eliminated will give rise to further subproblems by branching, so that the quality, or "tightness" of the bound will determine how much effort will be required to solve the problem.)

```
Branch-&-Bound Algorithm
0-1 Knapsack Problem

Randomly Generated Problem (seed 5354416)

>>>Subproblem # 1
J1:
J0:
JF:   1  2  3  4  5  6
Fractional solution: selected items = 1 3 5
                     plus 0.75 of item # 4
                     value = 35.25
Rounding down yields value 27
```

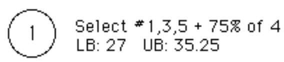
We begin with the original problem, calling it "subproblem" 1

By solving the LP relaxation, we get both upper & lower bounds

Notation:

J1 = indices of items forced into the knapsack ($X_j = 1$)
J0 = indices of items forced out of the knapsack ($X_j = 0$)
JF = indices of items free to be selected or rejected ($X_j \in \{0,1\}$)

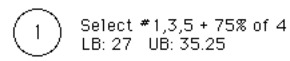
We will begin to construct a search tree, with a node representing subproblem 1:



We now have a feasible solution, with value 27, and we know that the optimal value cannot exceed 35.25

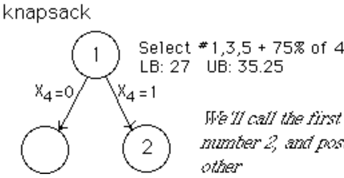
(Actually, since the values of the individual items are integer, we know that we cannot attain a value greater than 35!)

The feasible solution becomes our "incumbent" solution, the best solution known thus far, and the one for other candidate solutions to "beat"

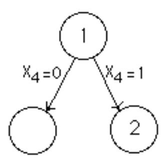


We will "branch" by creating two new subproblems, using item #4 as the "branching" variable:

- in one subproblem, item #4 is FORCED INTO the knapsack
- in the other subproblem, item #4 is FORCED OUT OF the knapsack



We'll call the first of these 2 subproblems number 2, and postpone numbering the other



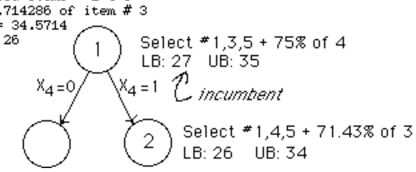
Clearly, either $X_4 = 1$ or $X_4 = 0$ in the optimal solution, so that the better solution of the two subproblems will be the solution to the original problem.

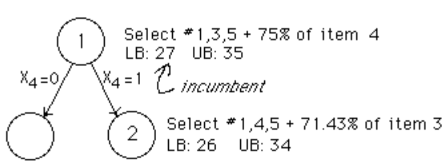
That is, if we find the best knapsack contents with the added restriction that we include item 4, and the best knapsack contents with the added restriction that we omit item 4, the optimal contents must be the better of these two.

We solve the LP relaxation of subproblem #2:

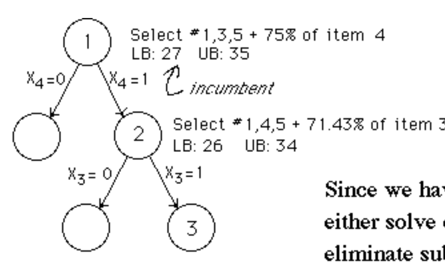
$$11 + \text{Max } 6X_1+10X_2+12X_3+9X_5+12X_6$$
$$\text{s.t. } 4X_1+17X_2+14X_3+9X_5+20X_6 \le 39-16=23$$
$$0 \le X_j \le 1, \; j=1,2,3,5,6$$

```
>>>Subproblem # 2
J1:   4
J0:
JF:   1  2  3  5  6
Fractional solution: selected items = 1 4 5
                     plus 0.714286 of item # 3
                     value = 34.5714
Rounding down yields value 26
```





At this time, we don't have the solution of either of the new subproblems, and since the upper bound of subproblem #2 is better than our incumbent (which is still the first incumbent with value 27), it is possible that subproblem #2 might yield a better optimal solution than the incumbent.



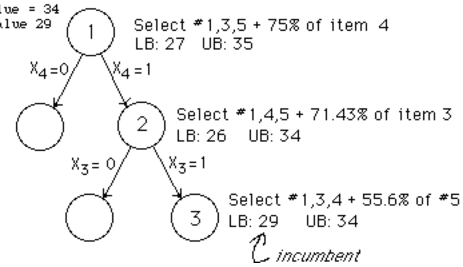
Note that in subproblem 3, BOTH items 3 and 4 are forced into the knapsack!

Since we haven't been able to either solve or otherwise eliminate subproblem #2, we again branch, by forcing item 3 either INTO or OUT OF the knapsack.

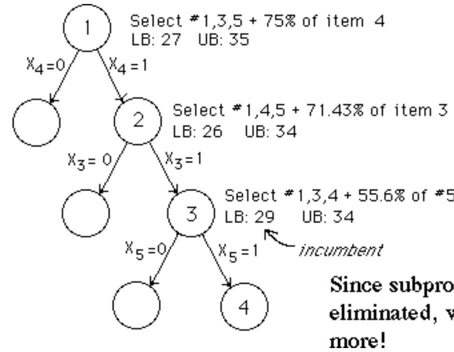
Solve the LP relaxation of subproblem 3:

$$23 + \text{Max } 6X_1+10X_2+9X_5+12X_6$$
$$\text{s.t. } 4X_1+17X_2+9X_5+20X_6 \le 39-16-14=9$$
$$0 \le X_j \le 1,\ j=1,2,5,6$$

```
>>>Subproblem # 3
J1:   3  4
J0:
JF:   1  2  5  6
Fractional solution: selected items = 1 3 4
                     plus 0.555556 of item # 5
                     value = 34
Rounding down yields value 29
*** NEW INCUMBENT! ***
```



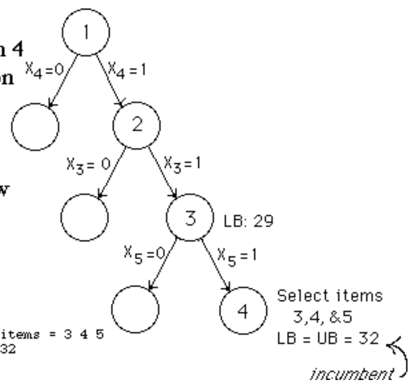
Notice that our incumbent has been replaced by a better feasible solution!



Since subproblem 3 isn't eliminated, we branch once more!

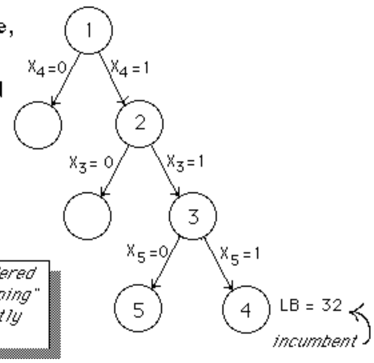
When we solve the LP relaxation of subproblem 4 we get an integer solution (which happens to be better than the old incumbent!) So subproblem #4 is now solved, and we need not branch further from it.

```
>>>Subproblem # 4
J1:   3  4  5
J0:
JF:   1  2  6
Integer solution: selected items = 3 4 5
                  Value= 32
*** NEW INCUMBENT! ***
```



We aren't finished, of course, since we still have three subproblems that we created and have not solved. Let's now consider the one most recently created, and call it subproblem #5:

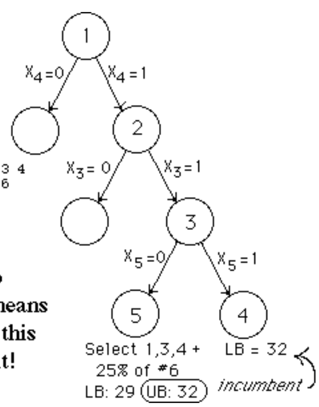
Any one of them could be considered next, but it simplifies "bookkeeping" to consider next the most recently created subproblem.



Solve the LP relaxation of subproblem #5:

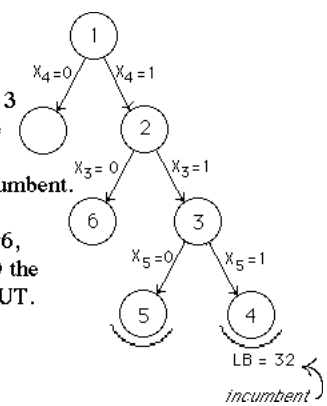
```
>>>Subproblem # 5
J1:   3  4
J0:   5
JF:   1  2  6
Fractional solution: selected items = 1 3 4
                     plus 0.25 of item # 6
                     value = 32
Rounding down yields value 29
<<<Subproblem # 5 fathomed.
```

Notice that the upper bound is no better than the incumbent; this means that we can eliminate ("fathom") this subproblem, and need not solve it!



Since both "descendants" (the two subproblems created from the subproblem) of subproblem 3 have been "fathomed", we have the optimum solution of subproblem #3, namely the incumbent.

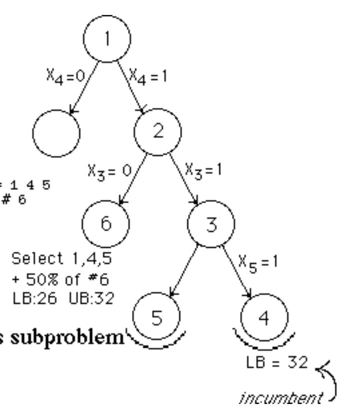
We next consider subproblem #6, which has item #4 forced INTO the knapsack and item #3 forced OUT.



Solve the LP relaxation of subproblem #6:

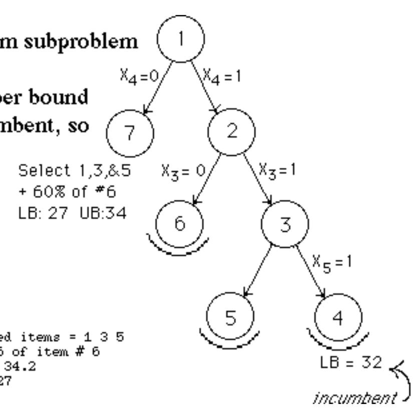
```
>>>Subproblem # 6
J1:   4
J0:   3
JF:   1  2  5  6
Fractional solution: selected items = 1 4 5
                     plus 0.5 of item # 6
                     value = 32
Rounding down yields value 26
```

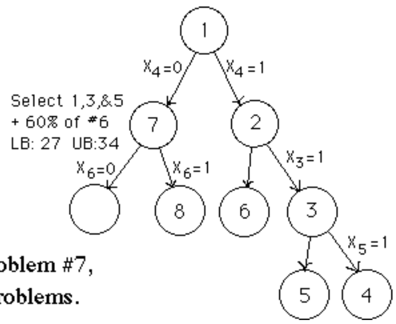
Again, because the upper bound is no better than the incumbent, we can fathom this subproblem



If we could now fathom subproblem #7, we'd be done. Unfortunately, it's upper bound is better than the incumbent, so the optimum of subproblem #7 might be optimal in the original problem!

```
>>>Subproblem # 7
J1:
J0:   4
JF:   1  2  3  5  6
Fractional solution: selected items = 1 3 5
                     plus 0.6 of item # 6
                     value = 34.2
Rounding down yields value 27
```



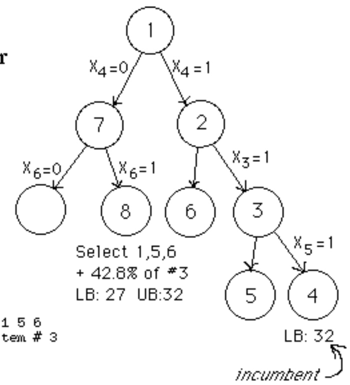


We branch from subproblem #7, creating two new subproblems.

Solving the LP relaxation of subproblem #8 yields an upper bound which is no better than the incumbent, so we can fathom the subproblem.

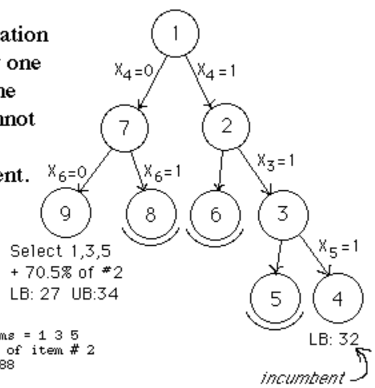
Remember, since the optimal value is integer, it can't be >32, although the LP solution is 32.14

```
>>>Subproblem # 8
J1:   6
J0:   4
JF:   1  2  3  5
Fractional solution: selected items = 1 5 6
                     plus 0.428571 of item # 3
                     value = 32.1429
Rounding down yields value 27
<<<Subproblem # 8 fathomed.
```

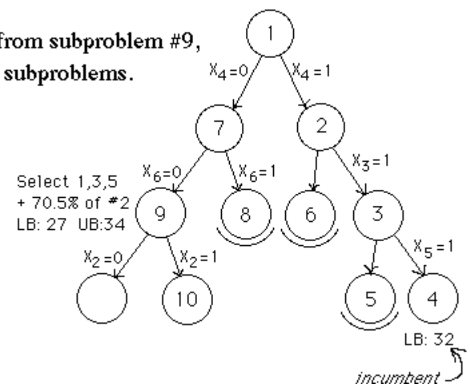


We next solve the LP relaxation of subproblem #9, the only one remaining unfathomed in the tree; unfortunately, we cannot fathom it, since the upper bound exceeds the incumbent.

```
>>>Subproblem # 9
J1:
J0:   4  6
JF:   1  2  3  5
Fractional solution: selected items = 1 3 5
                     plus 0.705882 of item # 2
                     value = 34.0588
Rounding down yields value 27
```

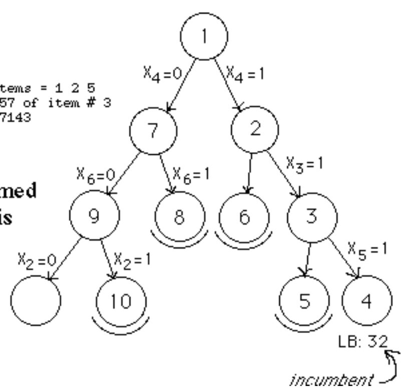


We must branch from subproblem #9, creating two new subproblems.




```
>>>Subproblem # 10
J1:   2
J0:   4  6
JF:   1  3  5
Fractional solution: selected items = 1 2 5
                     plus 0.642857 of item # 3
                     value = 32.7143
Rounding down yields value 25
<<<Subproblem # 10 fathomed.
```

Subproblem #10 is fathomed because its upper bound is no better than the incumbent.




```
>>>Subproblem # 11
J1:
J0:   2  4  6
JF:   1  3  5
Integer solution: selected items = 1 3 5
                  Value= 27
<<<Subproblem # 11 fathomed.
<<<Subproblem # 9 fathomed.
<<<Subproblem # 7 fathomed.
<<<Subproblem # 1 fathomed.
```

Finally, subproblem #11 is fathomed (since it has an integer solution, which is not as good as the incumbent). Since no subproblems remain, we are finished!