Wed, March 9 - Day 19

Topic 8 - Branch and Bound (continued)

General Procedure:

the search space or some subspace

(maximizing)

```
function bb(S, best_sol = None)
    if best_sol is None:
        best_score = -∞
    else:
        best_score = score(best_sol)

    if |S| = 1:
        candidate = the one thing in S
        value = score(candidate)
        if value > best_score:
            return candidate
    else:
        return best_sol
    # case where |S| > 1
    S1, S2 = branch(S)
    if bound(S1) > best_score:
        best_sol = bb(S1, best_sol)
    *   best_score = score(best_sol)
```

returns best sol in the subspace S

(the subspace only has a single sol, so we check it)

base case

if candidate satisfies the constraints

maybe we could do better

```
    if bound(S2) > best_score:
        best_sol = bb(S2, best_sol)

return best_sol
```

remove the "if"s, then you get backtracking

either the best_sol passed in, or the best solution in S, whichever is better

Relaxation

Let's try to figure out a bound for the knapsack problem.

Capacity: 14

item	weight	value
~~1~~	~~8~~	~~13~~
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

capacity: 11
value: 7

Branching: just like before
→ item 1 is in or out
→ item 2 is in or out

Bound: Suppose we have decided item 1 is out and item 2 is in.

How can we find an upper bound on the best we could possibly do with the rest of the solution?

* Greedy sol is not an UB, it's a LB.
* "Add up the value of everything remaining is technically an UB, but a not very good one.
* Computing the UB can't be too slow.

Capacity: 14

item	weight	value
~~1~~	~~8~~	~~13~~
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

The trick is relaxation. Sometimes it's easier to find an UB if you adjust the problem to be more permissible.

Fractional Knapsack: You are allowed to take fractions of items.

Capacity: 14

item	weight	value	take		
1	8	13	0.5	50%	4 / 6.5
2	3	7	1	100%	3 / 7
3	5	10	1	100%	5 / 10
4	5	10	0.4	40%	2 / 4
5	2	1			
6	2	1			
7	2	1			
					14 / 27.5

Theorem: A greedy and optimal solution to the Fractional Knapsack problem can be found by:

(1) Order the items by $\frac{\text{value}}{\text{weight}}$, decreasing

(2) take items from the top in full, until you can't anymore

(3) take whatever fraction of the next item that you can.

Capacity: 14

item	weight	value	density	order		
1	8	13	1.625	4	12.5%	13/8
2	3	7	2.333	1	100%	3
3	5	10	2	2	100%	5
4	5	10	2	3	100%	5
5	2	1	0.5	5		
6	2	1	0.5	6		28.625
7	2	1	0.5	7		

If capacity = 10, you get an optimal score of 21, which beats the optimal score of 20 for the regular knapsack problem with the same items.

Fractional Greedy = Fractional Optimal
$\geq$ Regular Optimal

Therefore, we can get an UB for the regular knapsack problem by computing the greedy fractional solution on the remaining items.

Capacity: 14

item	weight	value
~~1~~	~~8~~	~~13~~
2	3	7
3	5	10
4	5	10
5	2	1
6	2	1
7	2	1

capacity = 11
value = 7

5 items

densities

2	100%
2	100%
1/2	50%
1/2	
1/2	

capacity = ~~6~~ 1

value: $7 + 10 + 10 + \frac{1}{2} = 27.5$

Capacity = 10, B+B tree

Greedy sol: ~~18~~ 20 (most value-dense)