Branch and Bound

Design challenges:

- finding a good bound function
- finding a good initial solution value
- proving that clever bounds and initial solution values are safe

These tend to be highly problem-specific.

0-1 Knapsack

- frame as a series of choices – choose next item to take
- partial solution – set of items taken so far
- alternatives – items not yet chosen which fit in the pack
- subproblem – knapsack(S',W')
  - input: set S' of items left to consider, remaining unfilled capacity of the knapsack W'
  - output: items chosen, total value of those items
  - task: find the highest-value set of items whose total weight does not exceed the remaining capacity of the knapsack
- should we bother to solve knapsack(S',W')?
  - pruning: no, if the smallest item in S' exceeds W' nothing else will fit – the partial solution is actually a complete solution (treat as a base case)
  - branch and bound: no, if the best way to fill the pack from this point isn't better than the best solution already found
    - need an estimate of the best solution obtainable from this point
    - need an initial value for the best solution already found (until we find the first solution)

0-1 Knapsack

Bound function – upper bound or lower bound?

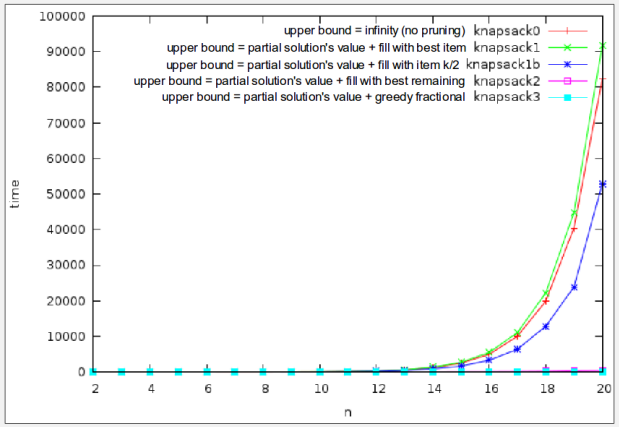
- maximization problem, so bigger is better
- prune if solutions aren't good enough → "safe" is an estimate which is too good → "too good" is bigger → looking for upper bound on the solution value

Which of the following would be safe choices for a bound function? Use the value of the items chosen so far plus ...

- ☐ filling the pack with the items will fit in their entirety (no fractional amounts), considered in order of decreasing value/weight ratio
- ☐ filling the pack with as much as possible of each of the remaining items (a fractional amount is allowed), in order of increasing value
- ☐ the lowest value/weight ratio of any item times the remaining capacity of the pack
- ☐ filling the pack with as much as possible of each of the remaining items (a fractional amount is allowed), in order of decreasing value
- ☐ the lowest value/weight ratio of any remaining (not yet considered) item times the remaining capacity of the pack
- ☐ filling the pack with as much as possible of each of the remaining items (a fractional amount is allowed), in order of increasing value/weight ratio
- ★ the highest value/weight ratio of any item times the remaining capacity of the pack
- ★ filling the pack with as much as possible of each of the remaining items (a fractional amount is allowed), in order of decreasing value/weight ratio
- ☐ filling the pack with the items will fit in their entirety (no fractional amounts), considered in order of increasing value/weight ratio
- ★ the highest value/weight ratio of any remaining (not yet considered) item times the remaining capacity of the pack

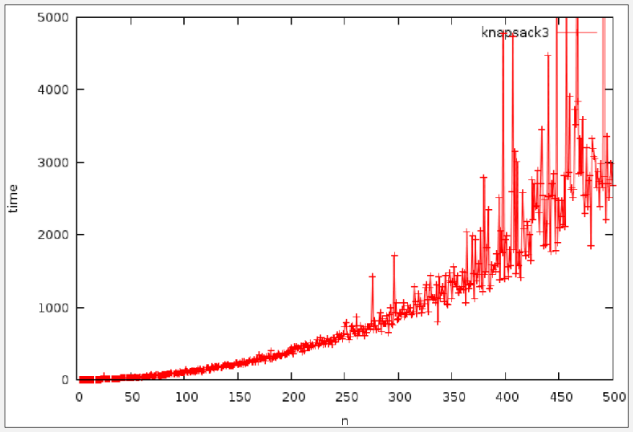
Bound Function Quality

range 1-100
capacity ~half of the total weight of all items



Bound Function Quality

(compare the range on the scales in this to the previous slide)



Strategies

A good bound function depends on the specific nature of the problem and what you can exploit about its structure.

But there are a few general tactics that might serve as starting points –

- value so far + best single choice × number of choices left
- value so far + best single next choice × number of choices left
  - only safe if all choices are available at each stage (e.g. knapsack but not TSP)
- value so far + greedy solution from that point
  - only safe if greedy can do better than the actual solution (true for knapsack, not for TSP and max independent set)
- consider trivial bound and what is over/undercounted
  - e.g. max independent set – |S| overcounts because a vertex and its neighbor can't both be in the set; |S|-mindeg(S) addresses that for one vertex picked

0-1 Knapsack

Initial solution value – upper bound or lower bound?

- maximization problem, so bigger is better
- update if solution is better → "safe" is an estimate which is not good enough → "not good enough" is smaller → looking for lower bound on the solution value

Which of the following would be safe choices for an initial solution estimate? Choose all that apply.

- ★ consider the items in any order, taking each item if it fits in the pack
- ★ 0
- ☐ the lowest value/weight ratio of any item times the capacity of the pack
- ☐ the highest value/weight ratio of any item times the capacity of the pack
- ★ consider the items in increasing order of value/weight ratio, taking each item if it fits in the pack
- ★ consider the items in decreasing order of value/weight ratio, taking each item if it fits in the pack

fill the pack with as much of each item as will fit (fractional amounts allowed), considering items in order of decreasing value/weight ratio

Initial Solution Estimate

Upper or lower bound?

- safe = conservative = worse than the optimal
  - if your estimate is better than the optimal, you'll prune away the branch containing the optimal as not good enough

Note: bound is on the value of the optimal solution, not the value of any legal solution

- e.g. “upper bound” does not mean that it needs to be worse than all possible legal solutions – and that wouldn't help you prune anything at all

Initial Solution Estimate

Any legal solution is a safe estimate – it will be no better than the optimal.

- greedy can be a good strategy
  - e.g. greedy TSP – take cheapest edge to not-yet-included vertex
  - e.g. maximal independent set – take any legal vertex until there are no more

But you may be able to get a tighter estimate without having an actual solution in mind.

(Then safety is important to establish.)

- e.g. 2*MST ≥ optimal TSP solution