Math 343, Lecture 20

① Bounding functions

The pruning which we've done so far still left us with a pretty slow knapsack algorithm. How can we do better?

Here is the general structure

Def Given a feasible solution X, let profit(X) be

Given a partial solution X
let $P(X)$ be

Then $P(\emptyset)$ is

It is too slow

So what we want is a bounding function

Idea

So

A good $B(X)$ is

-
-

Metaalgorithm Bounded Backtrack

global X, $OptP$, $OptX$, C_ℓ , $\ell = 0, 1, \ldots$

input ℓ

if $[x_0, \ldots, x_{\ell-1}]$ is feasible

Compute C_ℓ

B

for $x_\ell \in C_\ell$

Bounded Backtrack $(\ell+1)$

② Bounding for the Knapsack problem

What is a good bounding function for the Knapsack problem?

Consider a related problem

Rational Knapsack problem

Given profits $p_0, p_1, \ldots, p_{n-1}$
weights $w_0, w_1, \ldots, w_{n-1}$
and capacity M

Find

This problem is much easier to solve. A greedy algorithm will do

Algorithm Rational Knapsack

input: $p_0, p_1, \ldots, p_{n-1}, w_0, \ldots, w_{n-1}, M$

permute

$c = 0$

$P = 0$

$W = 0$

$X = [0, 0, \ldots 0]$

$x_0 \; x_1 \cdots x_{n-1}$

while

This algorithm runs in

We can use the rational knapsack problem to give a bounding function for our original knapsack problem

For a partial solution $X = [x_0, x_1, \ldots, x_{k-1}]$ define

$B(X) =$

So we have the following algorithm

Algorithm Bounded Knapsack

global X, $OptX$, $OptP$, C_ℓ $\quad \ell = 0, 1, \ldots$

assume the p_i, w_i are ordered so $\frac{p_0}{w_0} \geq \frac{p_1}{w_1} \geq \ldots \geq \frac{p_{n-1}}{w_{n-1}}$

input ℓ, $CurW$

if $\ell = n$

else

$B =$

for $x_\ell \in C_\ell$

BoundedKnapsack$(\ell+1, CurW + w_\ell x_\ell)$

eg $p_0=23,\ p_1=29,\ p_2=15,\ p_3=13,\ p_4=16$

$w_0=11,\ w_1=12,\ w_2=8,\ w_3=7,\ w_4=9$

$n=5,\ M=26$

if $l=n$

if $\sum_{i=0}^{n-1} p_i x_i > Opt\,P$

$Opt\,P = \sum_{i=0}^{n-1} p_i x_i$

$Opt\,X = [x_0, \ldots, x_{n-1}]$

$C_l = \phi$

else

if $Cur\,W + w_l \le M$

$C_l = \{0,1\}$

else

$C_l = \{0\}$

$B = \sum_{i=0}^{l-1} p_i x_i$ + Rational Knapsack

$(p_l, \ldots, p_n, w_l \ldots, w_n, M - Cur\,W)$

for $x_l \in C_l$

Bounded Knapsack $(l+1,$

$Cur\,W + w_l x_l)$

Hoe is some experimental runtime data from Kreher and Stinson p127

The instances were generated by for each i randomly selecting w_i an integer between 0 and 1000 000 and then choosing $p_i = 2w_i\varepsilon_i$ whoe ε_i is random in the interval (0.9, 1.1) M is $\frac{1}{2}\sum_{i=0}^{n-1} w_i$

TABLE 4.2
Size of state space trees for Algorithms 4.1, 4.3, and 4.9, on random instances with n weights

n	Naive Knapsack Algorithm 4.1	Pruned Knapsack Algorithm 4.3	Bounded Knapsack Algorithm 4.9
8	511	332	52
8	511	312	78
8	511	333	72
8	511	321	74
8	511	313	57
12	8191	4598	109
12	8191	4737	93
12	8191	5079	164
12	8191	4988	195
12	8191	4620	87
16	131071	73639	192
16	131071	72302	58
16	131071	76512	168
16	131071	78716	601
16	131071	78510	392
20	2097151	1173522	299
20	2097151	1164523	104
20	2097151	1257745	416
20	2097151	1152046	118
20	2097151	1166086	480
24	33554431	19491410	693
24	33554431	18953093	180
24	33554431	17853054	278
24	33554431	19814875	559
24	33554431	18705548	755

③ Traveling salesman problem

This is another classic problem in algorithms and optimization

Suppose there are

We might as well

So the formal problem statement is

Travelling salesman problem

Given

Find

Let the vertex set of G be $V = \{0, \dots, n-1\}$

We can

Algorithm Naive Travelling Salesman

global X, $OptC$, $OptX$, C_ℓ $\ell = 0, 1, \dots$

input ℓ

if $\ell = n$

if $\ell = 0$

if $\ell = 1$

else

for $x_\ell \in C_\ell$

NaiveTravelling Salesman $(\ell+1)$

Now lets do better with a bounding function
In this problem

Def Given $x \in V$, $\Omega \subseteq V$, $\Omega \neq \emptyset$ define

$$b(x, \Omega) =$$

Proposition Let $X' = [x_0, \ldots, x_{n-1}]$ be the minimum cost Hamiltonian cycle (given as a list of vertices) which extends $[x_0, \ldots, x_{\ell-1}]$ $(\ell < n)$. Then

$$\text{cost}(X') \geq \sum_{i=0}^{\ell-2} \text{cost}(x_i, x_{i+1}) + b(x_{\ell-1}, Y) + \sum_{y \in Y} b(y, Y \cup \{x_0\})$$

where $Y = V \setminus \{x_0, \ldots, x_{\ell-1}\}$

proof For convenience let $x_n = x_0$

$$\text{cost}(X') =$$

This bounding function is called the min cost bound

It gives the following algorithm

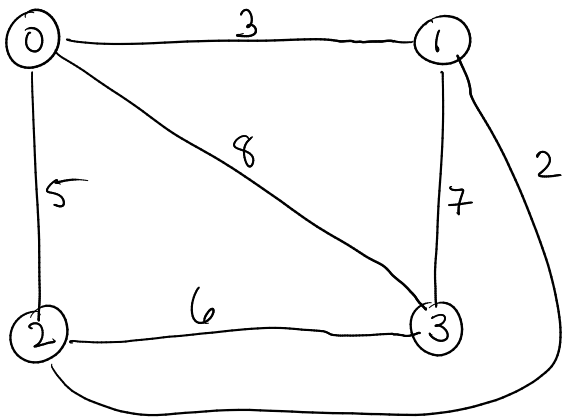
Algorithm MinCostBoundTravellingSalesman

global $X, OptC, OptX, C_\ell$ $\ell = 0, 1, \ldots$

input ℓ

Min Cost Bound Travelling Salesman (l+1)

eg Consider the following graph and costs



Lets run the min cost bound travelling salesman on this graph

if $l = n$

$\hat{C} = \text{cost}([x_0, \ldots, x_{n-1}])$

if $C < \text{Opt } C$

$\text{Opt } C = C$

$\text{Opt } X = [x_0, \ldots, x_{n-1}]$

return

if $l = 0$

$C_l = \{0\}$

else

if $l = 1$

$C_l = \{1, \ldots, n-1\}$

else

$C_l = C_{l-1} \setminus \{x_{l-1}\}$

$$B = \sum_{i=0}^{l-2} \text{cost}(x_i, x_{i+1}) + b(x_{l-1}, Y) + \sum_{y \in Y} b(y, Y \cup \{x_0\})$$

if $B \geq \text{Opt } C$

return

for $x_l \in C_l$

Min Cost Bound Travelling Salesman $(l+1)$

IS

④ Next time

Better bounds for Travelling Salesman

Branch and bound