27 April 2018

Greedy approx alg for load balancing

Initialize $J_i = \emptyset$, $L_i = 0$ $\forall i = 1, \dots, m$

for $j = 1, 2, \dots, n$

  assign j to machine i with smallest load L_i.
  (break ties arbitrarily.)

  $J_i \leftarrow J_i \cup \{j\}$

  $L_i \leftarrow L_i + s_j$

endfor

$(m=2)$

Example where this is suboptimal: $s_1 = \frac{1}{2}$ $s_2 = \frac{1}{2}$ $s_3 = 1$.

$OPT = 1$ $J_1 = \{1,2\}$, $J_2 = \{3\}$

Greedy instead does:

Iteration	Machine 1	Machine 2
1	$1/2$	
2		$1/2$
3	1	

Ends with loads $\{3/2, 1/2\}$ $ALG = 3/2$ while $OPT = 1$.

Theorem. For all input instances, $GREEDY \leq 2 \cdot OPT$.

Proof. Suppose machine i is the max loaded machine when you run GREEDY, and j is the last job that gets put into J_i.
Let L be the load on machine i just before j was assigned to it. So...

$$\boxed{GREEDY = L + s_j}$$

On the other hand $\boxed{OPT \geq s_j}$ because job j has to be assigned somewhere in OPT. And also $\boxed{OPT \geq L}$, because

max of m numbers is $\geq$ their average

$$OPT \geq \frac{1}{m} \sum_{k=1}^{n} s_k \geq L$$

At the moment j is assigned to i, each machine has load $\geq L$ so combined job size is $\geq mL$.

Combining the equations in pink boxes...

$$OPT \geq s_j \,, \qquad OPT \geq L$$

$$\therefore \quad 2\,OPT \geq s_j + L = GREEDY$$

QED.

Improved greedy algorithm: initially sort jobs in decreasing order of size.

<u>Theorem.</u> If $s_1 \geq s_2 \geq \cdots \geq s_n$ and we assign jobs using GREEDY,

$$GREEDY \leq \frac{3}{2}\, OPT.$$

<u>Proof.</u> Define i, j as in preceding proof. Define L as before.

$$GREEDY = L + s_j$$

As before, $OPT \geq L$.

If $L = 0$, then $OPT \geq s_j = GREEDY$, and we're done in this case.
If $L > 0$ it means every machine has positive load before j is assigned.
$\therefore$ $j \geq m+1$. $\therefore$ $\{s_1, \ldots, s_j\}$ is a set of at least $m+1$ job sizes, the smallest of which is s_j. Pigeonhole $\Rightarrow$ OPT must assign at least two jobs of size $\geq s_j$ to the same machine.

$$OPT \geq 2 s_j.$$

$$\frac{1}{2}\, OPT \geq s_j \,, \qquad OPT \geq L$$

$$\frac{3}{2}\, OPT \geq s_j + L = GREEDY.$$

Designing & Analyzing Approx Algs Using Dynamic Programming (§11.8)

This method often works for problems that are NP-Hard because the input data includes high-precision numbers, i.e. problems that have pseudopolynomial algorithms such as Knapsack.

<u>RECAP:</u> Items $1, \ldots, n$ size s_i, value v_i (pos integers)
Overall size budget B.
Find a subset $S \subseteq \{1, \ldots, n\}$ to maximize $\sum_{i \in S} v_i$ subject to $\sum_{i \in S} s_i \leq B$.
Assume $s_i \leq B \;\; \forall i$. (Deleting i with $s_i > B$ doesn't affect the answer.)

Pseudopolynomial algorithm was a dynamic program:

Dyn prog table stored $T[i,j] \triangleq$ max combined value of a subset of $\{1,\dots,i\}$ whose combined size is $\le j$.

$0 \le i \le n, \quad 0 \le j \le B.$

for $i = 0, \dots, n$
  for $j = 0, \dots, B$
    if $i = 0$ or $j = 0$ $\quad T[i,j] = 0.$
    else if $s_i > j$ $\quad T[i,j] = T[i-1,j]$
    else $\quad T[i,j] = \max\{T[i-1,j],\ T[i-1,j-s_i] + v_i\}$
  endfor
endfor

if B is n digits long, its magnitude could be $2^{O(n)}$.

Running time: $\quad O(n \cdot B).$

Create a modified knapsack instance where each item i has size $\tilde{s}_i := \lceil \varepsilon s_i \rceil$. ε = a small positive number, value TBD. (Value of ε will be specified at end of designing alg, at the start of running it. Doing the analysis in terms of a parameter ε and setting ε ~~at~~ the end to make it all work is a common trick.)

ABORT.

A different dyn prog. $\quad U[i,j] \triangleq$ minimum combined size of a subset of $\{1,\dots,i\}$ whose combined value is $\ge j$.

$0 \le i \le n, \quad 0 \le j \le V \triangleq \sum_{i=1}^{n} v_i.$

for $i = 0, \dots, n$
  for $j = 0, \dots, V$
    if $i = 0$ $\quad U[i,j] = \begin{cases} 0 & \text{if } j = 0 \\ \infty & \text{if } j > 0. \end{cases}$
    if $j = 0$ $\quad U[i,j] = 0$
    else $\quad U[i,j] = \min\{U[i-1,j],\ s_i + U[i-1, \max\{0, j-v_i\}]\}$
  endfor
endfor

Output $\max\{j \mid U[n,j] \le B\}.$

Running time $\quad O(n \cdot V).$

Modify the problem by $\tilde{v}_i = \lceil \varepsilon \cdot v_i \rceil$.
Keep same s_i, B.

Solve knapsack using the 2nd algorithm above.
Runs in time $O(n \cdot \sum \tilde{v}_i) = O(n \cdot \sum_{i=1}^{n} (\varepsilon v_i + 1)) = O(n^2 + \varepsilon n V)$.

How close to OPT?