

APPROXIMATION ALGORITHMS

An approximation algorithm is a polynomial algorithm, evaluated by its worst case performance.

An algorithm A is a δ -APPROXIMATION ALGORITHM for a minimization problem P if for every instance I of P it delivers a solution that is at most δ times the optimum.

Eg: Christofides heuristic for TSP problem satisfying the triangle is a ~~$\frac{3}{2}$ -approx~~ $\frac{3}{2}$ -approx algo.

BIN PACKING (Coffman, Henry Johnson chapter 2 in "Approx Algor for NP-Hard Problems", edited by Hochbaum, PWS, 1997.)

Given a sequence $L = (a_1, a_2, \dots, a_n)$ of items, each with size $s(a_i) \in (0, 1]$, and want to pack them into a minimum number of bins, each with capacity 1.

Applications:

- loading trucks subject to weight limitations
- packing television commercials into station breaks
- stock-cutting problems

...

Worst Case Analysis:

For a given list L and algorithm A , let $A(L)$ be the number of bins used when algorithm A is applied to list L .

Let $OPT(L)$ denote optimum number of bins for a packing of L

Let $R_A(L) := A(L) / OPT(L)$.

Absolute worst case performance ratio:

$R_A := \inf \{ r \geq 1 : R_A(L) \leq r \text{ for all lists } L \}$ (we don't use this again)

Asymptotic worst case performance ratio:

$R_A^\infty := \inf \{ r \geq 1 : \text{for some } N > 0, R_A(L) \leq r \text{ for all } L \text{ with } OPT(L) \geq N \}$.

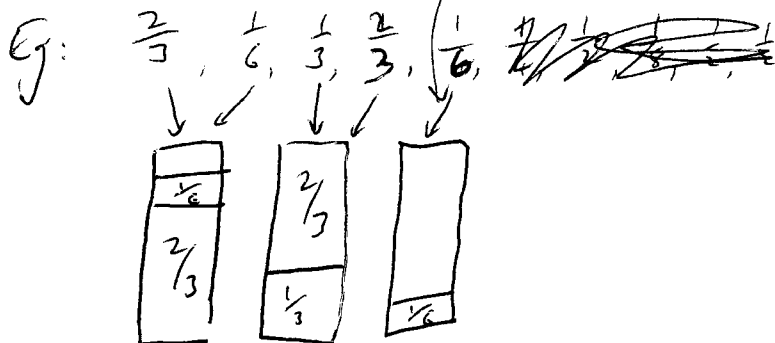
~~Next Fit Heuristic:~~

First consider ONLINE case, where the items arrive one at a time and have to be assigned to a bin immediately:

Next fit Heuristic:

The only partially filled bin that is open is the most recent one to be packed.

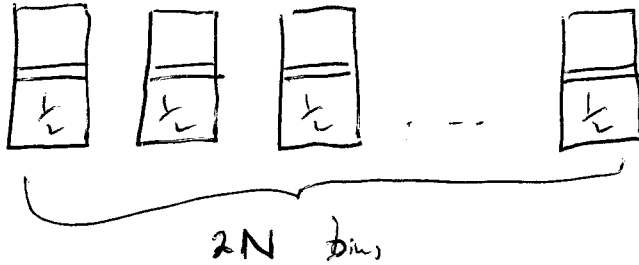
note: not allowed to assign this to the first bin.



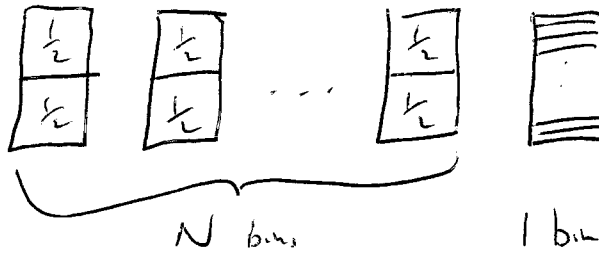
Bad case for next bit:

$$L = \left(\frac{1}{2}, \frac{1}{2N}, \frac{1}{2}, \frac{1}{2N}, \frac{1}{2}, \dots, \frac{1}{2}, \frac{1}{2N} \right) \quad 4N \text{ items.}$$

Get assignment:



Optimal assignment:



$$\frac{A(L)}{\text{OPT}(L)} \approx 2.$$

Can also show that $A(L) \leq 2 \text{OPT}(L) - 1$

(since the total use of each bin and the one that follows it must be > 1 .)

Thus, $R_{\text{FA}}^{\infty} = 2.$

First Fit Heuristic

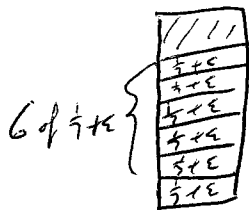
All partially filled bins are acceptable.

For all lists L , $FF(L) \leq \lceil 1.7 OPT(L) \rceil$.

Bad list: ~~$(\frac{1}{7}, \frac{1}{7}, \dots, \frac{1}{7}, \frac{1}{6}, \dots, \frac{1}{5}, \frac{1}{2}, \dots, \frac{1}{2})$~~

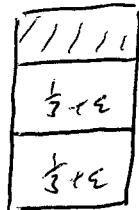
$(\underbrace{\frac{1}{7} + \epsilon, \frac{1}{7} + \epsilon, \dots, \frac{1}{7} + \epsilon}_{6N \text{ terms}}, \underbrace{\frac{1}{5} + \epsilon, \dots, \frac{1}{5} + \epsilon}_{6N \text{ terms}}, \underbrace{\frac{1}{2} + \epsilon, \dots, \frac{1}{2} + \epsilon}_{6N \text{ terms}})$

First fit gives:



6 of $\frac{1}{7} + \epsilon$

N bins



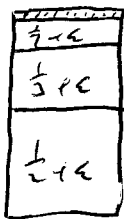
$3N$ bins



$6N$ bins

Total: $10N$ bins.

Optimal:



$6N$ bins

Thus, $\frac{FF(L)}{OPT(L)} = \frac{10}{6} = 1.66\dots$

Can extend this construction to get $\frac{FF(L)}{OPT(L)} \approx 1.69103$.

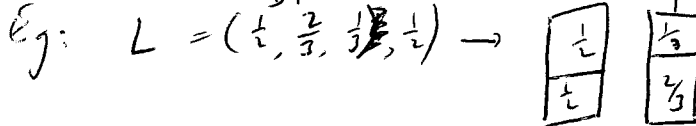
More complicated constructions give $FF(L) > \frac{17}{10} OPT(L) - 2$. So $R_{FF}^{\infty} = 1.7$

Best fit:

Item a_j is packed in ~~last~~ the partially filled bin with the highest level into which it will fit.

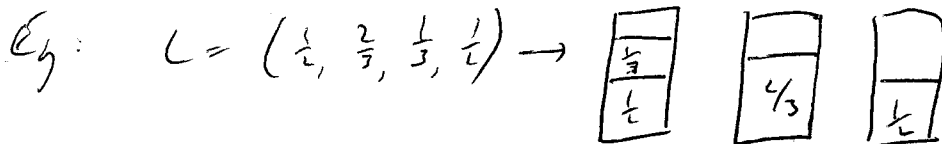
Can get $BF(L) = \frac{4}{5} FF(L)$ for one list L
 and $FF(L) = \frac{7}{2} BF(L)$ for another list L .

Asymptotically, R_{BF}^{∞} behaves like R_{FF}^{∞} .



Worst fit:

Item a_j is packed in the partially filled bin with the lowest level, and otherwise starts a new bin.



Asymptotically: $R_{WF}^{\infty} = R_{NF}^{\infty}$. (Use same example as for NF.)

Offline Algorithms:

Knows all ~~objects~~ ^{items} before we pack any of them.

Suggests sorting items in decreasing order of size.

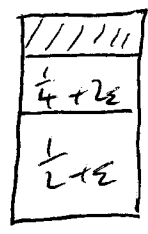
Leads to First Fit Decreasing & Best Fit Decreasing algorithms:

Thm. $R_{FFD}^{\infty} = R_{BFD}^{\infty} = 11/9.$

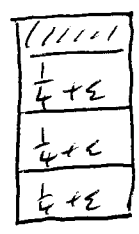
Bad example:

$$L = (\underbrace{\frac{1}{2} + \epsilon, \dots, \frac{1}{2} + \epsilon}_{6N \text{ items}}, \underbrace{\frac{1}{4} + 2\epsilon, \dots, \frac{1}{4} + 2\epsilon}_{6N \text{ items}}, \underbrace{\frac{1}{4} + \epsilon, \dots, \frac{1}{4} + \epsilon}_{6N \text{ items}}, \underbrace{\frac{1}{4} - 2\epsilon, \dots, \frac{1}{4} - 2\epsilon}_{12N \text{ items}})$$

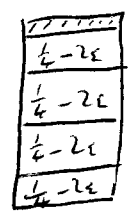
Algorithms give:



6N bins

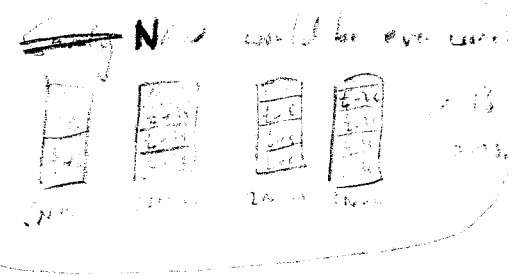


2N bins

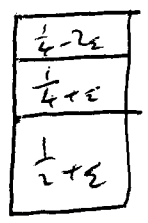


3N bins

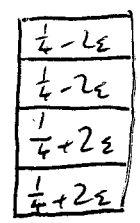
So use 11N bins.



Optimal soln:



6N bins



3N bins

So use 9N bins.

Thus, $\frac{FFD(L)}{OPT(L)} = \frac{BFD(L)}{OPT(L)} = \frac{11}{9}.$

Thm : For any $\epsilon > 0$, there exists a linear time algorithm A_ϵ such that

$$R_{A_\epsilon}^\infty \leq 1 + \epsilon.$$

Idea : peel off small elements in linear time,
find a good packing with remaining items,
stick small items back in.

A PROXIMATION ALGOS FOR SET COVERING. (Kocubani, Chapter 3 of his book)

Defn. A Vertex Cover is $G = (V, E)$ // set of vertices C such that each edge

Greedy Algo for Set Cover:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq e \\ & x \text{ binary.} \end{aligned}$$

columns \leftrightarrow sets
rows \leftrightarrow items

- Applications:
- efficient testing
 - statistical design of experiments
 - crew scheduling for airlines
 - ...

Pick At each step, pick index j which minimizes the ratio $c_j / |S_j|$ where $|S_j|$ is the number of currently uncovered ~~items~~ that ~~are in set~~ j .

Eg:

$$\begin{aligned} \min \quad & x_1 + 3x_2 + 2x_3 + 2x_4 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \\ & x_2 + x_3 \geq 1 \\ & x_3 + x_4 \geq 1 \\ & x_1 + x_4 \geq 1 \\ & x_i \text{ binary} \end{aligned}$$

First pick x_1 . Left with:

$$\begin{aligned} \min \quad & 3x_2 + 2x_3 + 2x_4 \\ \text{s.t.} \quad & x_2 + x_3 \geq 1 \\ & x_3 + x_4 \geq 1 \\ & x_i \geq 1. \end{aligned}$$

So next pick x_3 . Done.

Eg: $\max 4x_1 + 5x_2 + 7x_3 + 9x_4 + 8x_5 + 3x_6$

st. $x_1 + x_6 \geq 1$
 $x_1 + x_2 + x_4 \geq 1$
 $x_1 + x_3 + x_5 \geq 1$
 $x_2 + x_5 \geq 1$
 $x_3 + x_4 \geq 1$
 $x_i \geq 0$

Dual: $\max y_1 + y_2 + y_3 + y_4 + y_5$

st. $y_1 + y_2 + y_3 \leq 4$
 $y_2 + y_4 \leq 5$
 $y_3 + y_5 \leq 7$
 $y_2 + y_5 \leq 9$
 $y_3 + y_4 \leq 8$
 $y_i \geq 0$

Greedy: Pick x_1 . $|S_1| = 3$.
 So set $y_1 = y_2 = y_3 = 4/3$.
 (= average price paid by x to cover first three items)

Pick x_2 . $|S_2| = 1$
 So set $y_4 = 5/1$

Pick x_3 , $|S_3| = 1$
 So set $y_5 = 7/1$

How good is this algorithm?

Recall definition of δ -Approx algo for page Approx 1.

Let $d =$ size of largest set
 $= \max \#$ of ones in a ~~row~~^{column} of A .

$$\text{Let } H(d) = \sum_{i=1}^d \frac{1}{i} \leq 1 + \log d \leq 1 + \log(m).$$

Then The greedy heuristic is an $H(d)$ -approximation algorithm.

Proof via LP duality:

~~Primal~~ problem. Primal problem: $\max c^T x$
 s.t. $Ax \geq e$
 $x \geq 0$

max $e^T y$
 s.t. $A^T y \leq c$
 $y \geq 0$

Find a dual solution that is sufficiently close to H in value to the greedy solution.

When we pick x_j , set $y_k = \frac{c_j}{|S_j|}$ for each $k \in S_j$.

Note that $\sum g_j = 3(\frac{4}{3}) + 5 + 7 = 16 =$ value of greedy soln.

But y is not dual feasible.

Can show that $\frac{y}{H(d)}$ is dual feasible:

Here, $H(d) = 1 + \frac{1}{2} + \frac{1}{3} = \frac{11}{6}$.

So take $y_1 = y_2 = y_3 = (\frac{4}{3})(\frac{6}{11}) = \frac{8}{11}$
 $y_4 = 5(\frac{6}{11}) = \frac{30}{11}$
 $y_5 = 7(\frac{6}{11}) = \frac{42}{11}$

This is dual feasible.

So value of greedy $\leq H(d)$.
lower bound

(To show dual feasibility in general, the notation gets messy.

Let S^i be the set of dual variables fixed at stage i

Let \hat{S}^i be the dual variables that appear in the constraint of interest and also appear in S^i

Look at original values (prior to scaling).

By the choice of the greedy approach,

$$\sum_{j \in \hat{S}^i} y_j \leq \frac{|\hat{S}^i| c_k}{|\hat{S}^i| + \text{size of remainder of this constraint } k} \quad \text{for constraint } k.$$

$$\leq c_k / (1 + \text{size of remainder})$$

Worst case: peel off one at a time. Then $\sum_{j \in \hat{S}^i} y_j \leq c_k (1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{a}) = c_k H(d)$.

Bad case for greedy: (Make this a homework exercise?)

$$\begin{array}{l}
 \text{min} \quad (1-\epsilon)x_1 + x_2 + x_3 \\
 \text{s.t.} \quad x_1 + x_2 \geq 1 \\
 \quad \quad x_1 + x_3 \geq 1 \\
 \quad \quad \quad x_2 \geq 1 \\
 \quad \quad \quad \quad x_3 \geq 1 \\
 \quad \quad x_i \geq 0
 \end{array}$$

$$\begin{array}{l}
 \text{max} \quad y_1 + y_2 + y_3 + y_4 \\
 \text{s.t.} \quad y_1 + y_2 \leq 1-\epsilon \\
 \quad \quad y_1 + y_3 \leq 1 \\
 \quad \quad \quad y_2 + y_4 \leq 1 \\
 \quad \quad y_i \geq 0
 \end{array}$$

Greedy gives $x_1 = 1 \Rightarrow y_1 = y_2 = \frac{1}{2}(1-\epsilon)$
 then $x_2 = 1 \Rightarrow y_3 = 1$
 and $x_3 = 1 \Rightarrow y_4 = 1$

Primal value: $3-\epsilon$, compared with optimal value of 2 ($x_1=0, x_2=x_3=1$).

For dual, have $y_1 + y_3 = 1 + \frac{1}{2}(1-\epsilon) \neq 1$.
 So divide by $1 + \frac{1}{2}(1-\epsilon) = \frac{3}{2} - \frac{1}{2}\epsilon = \frac{1}{2}(3-\epsilon) = H(d)$
 since $d=2$.

LP algorithm for set cover.

1. Solve LP relaxation of the set covering problem, get cols x^* , dual soln y^* .
2. Output the cover $C_{LP} = \{j: x_j^* > 0\}$.

This is a cover:

Know $\sum_{j \in S_i} x_j^* \geq 1$ for each constraint i , so increasing x_j can only help.

Value^{dlp} is $\sum_{j=1}^n c_j x_j^* = \sum_{i=1}^m y_i^*$.

Now, by complementary slackness, if $x_j^* > 0$ then $\sum_{i: j \in S_i} a_{ij} y_i^* = c_j$.

Thus, value of the cover is

$$\sum_{j: x_j^* > 0} c_j = \sum_{j: x_j^* > 0} \left(\sum_{i=1}^m a_{ij} y_i^* \right) = \sum_{i=1}^m y_i^* \left(\sum_{j: x_j^* > 0} a_{ij} \right)$$

the number of sets containing the most common item.

$$\leq g \sum_{i=1}^m y_i^* \quad \text{when } g = \text{size of largest set}$$

$$\leq g \cdot \text{OPT}$$

since $\sum_{i=1}^m y_i^*$ is a lower bound on optimal value OPT of the integer program.

Note: Should prune the resulting cover of unnecessary x_j , to get a minimal cover

Note: LP gives $x_1 = x_2 = x_3 = 1$, optimal, for earlier problem. ($y_1 = 3, y_2 = y_3 = 0, y_4 = 5, y_5 = \dots$ and others zero)

Bad examples for LP:

$$\begin{aligned} \min \quad & x_1 + x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 \geq 1 \\ & x_2 + x_3 \geq 1 \\ & x_1 + x_3 \geq 1 \\ & x_i \geq 0 \end{aligned}$$

Goal: ...

$$\begin{aligned} \min \quad & x_1 + \dots + x_n \quad (n \text{ odd}) \\ \text{s.t.} \quad & x_i + x_{i+1} \geq 1 \quad i=1, \dots, n-1 \\ & x_1 + x_n \geq 1 \\ & x_i \geq 0 \end{aligned}$$

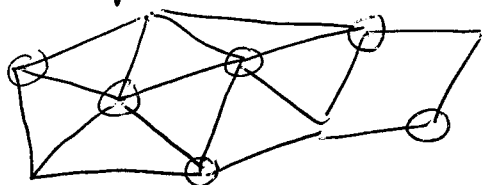
$x_i = 1/2$ for LP \Rightarrow value $n/2$ for ILP.
 True ILP value $n+1$ (\dots $\{1$ $i \text{ odd}$)

Vertex cover problem:

Given graph $G=(V,E)$, find a ~~subset~~ minimum subset of the ~~edges~~ vertices such that every edge is adjacent to at least one of the vertices.

Can assign weights w_v to the vertices.

Eg:



Cover (may not be optimal.)

This is a set covering problem:

$$\min \sum_{i=1}^n x_i$$

$$\text{s.t. } x_i + x_j \geq 1 \quad \forall \text{ edges } (i,j)$$

x_i binary.

Greedy algo gives $\frac{1}{2}d$ -approx scheme, where $d = \max$ degree of a vertex.

LP relaxation gives 2-approx scheme.

Another algo:

Find a maximal matching.

Take the ~~endpoints~~ endpoints of each edge in the matching.

This is a cover: Otherwise, another edge can be added to the matching.

It is a 2-approx algo:

If matching has $|M|$ edges, need at least $|M|$ vertices in the cover just to cover these edges.

We have $2|M|$ edges.

(Korte & Neumaier, 1978) basic feasible

Then In the solution to the LP-relaxation of the vertex cover, $x_v^{LP} = 0, \frac{1}{2}, 1$ for every vertex. Further there exists an optimal solution to the vertex cover problem with $x_v = \begin{cases} 1 & \text{if } x_v^{LP} = 1 \\ 0 & \text{if } x_v^{LP} = 0 \end{cases}$ (PERSISTENCE)

Can you use this to preprocess:

Solve LP relaxation, get x^{LP} .

Get the

Partition vertices into 3 sets:

- $j \in P$ if $x_j^{LP} = 1$
- $j \in Q$ if $x_j^{LP} = \frac{1}{2}$
- $j \in R$ if $x_j^{LP} = 0$.

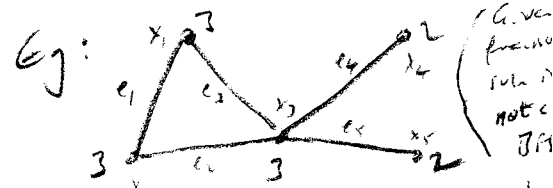
At least one optimal cover contains P.

Every vertex in R must ~~be adjacent to~~ have all its neighbours in P.
 Each cover in Q has weight at least $w(P) + \frac{1}{2}w(Q)$, the value of the LP relaxation.

~~Each cover has weight~~

The optimal cover has weight at most $w(P) + w(Q)$.

Thus: we can just try to find the best vertex cover in the graph $(Q, E(Q))$, i.e., the subgraph induced by Q.



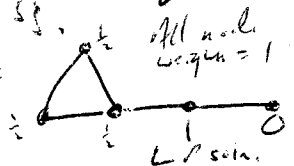
Eg: LP relax has max value, including $x_1 = x_2 = \frac{1}{2}, x_3 = 1, x_4 = 0, x_5 = 0$, value 6.

Dual optimal soln: $y_1 = 3, y_2 = y_3 = 0, y_4 = y_5$

Optimal vertex covers:

$\{1, 3\}$, or $\{2, 3\}$.

Better example:



All nodes in Q have weight = 1
 LP soln.

→ Pulleyblank (1979): Almost all randomly generated graphs have an LP-relaxation for which the solution is a vector of $\frac{1}{2}$'s, and no integer entries.

Then for I with 2 variables per row, the solution a 2-approximation solution can be found,

HARDNESS OF APPROXIMATIONS (CHAPTERS ~~20~~⁹⁻¹⁰ IN APPROX ALGOS BOOK)

Very hard to get good approximate solutions to some problems.

Eg: TSP: Consider Hamiltonian Circuit problem, on a graph $G = (V, E)$.
Give each edge in E weight 0.

(Alternatively:
neg. edge lengths
to 1 and $r|V|$,
so tour will have
length $|V|$
or $\geq r(n-1)$.)

Give each edge ~~to~~ not in E weight 1.

Then any Hamiltonian circuit for original graph has length 0, any other tour has length ≥ 1 .

If we can get ~~tour~~ within a factor of r , ~~then~~ in poly time, for TSP, then we can find a tour of length $\leq r \cdot \text{OPT}(I) = 0$ if \exists Hamiltonian circuit, and length ≥ 1 otherwise.

So, we could solve HC in poly time..

See p. 16A

DEFN A family of approximation algorithms for a problem P , $\{A_\epsilon\}_\epsilon$, is a POLYNOMIAL APPROXIMATION SCHEME, or PAS, if A_ϵ is a $(1+\epsilon)$ -approx alg and its running time is polynomial in the size of the input for a fixed ϵ .
(Note: running time may be exponential in $\frac{1}{\epsilon}$.)

DEFN A family of approximation algorithms ~~for~~^{exists} for a problem P is a FULLY POLYNOMIAL APPROXIMATION SCHEME, or FPAS, if A_ϵ is a $(1+\epsilon)$ -approx alg and its running time is polynomial in the size of the input and $\frac{1}{\epsilon}$.

THM (Garey & Johnson) There is no ~~fully polynomial~~ FPAS for a strongly NP-complete problem, unless $P=NP$.

Return to TSP:

Let L be the longest edge length in an instance of TSP.
Add L to each edge length.

Now the \triangle -ineq holds, since $c_{ij} + c_{jk} \geq 2L$, and $c_{ik} \leq L + L$.
Also, the length of every tour has increased by nL (not $n+1$), so
the best tour is still the best tour.

So we can use the Christofides heuristic.
Get

But, for the original lengths:

$$\begin{aligned} V_{\text{new}}(\text{CH}) &\leq \frac{3}{2} V_{\text{new}}^* \\ V_{\text{old}}(\text{CH}) &= V_{\text{new}}(\text{CH}) - nL \\ &= \frac{3}{2} V_{\text{new}}^* - nL \\ &= \frac{3}{2} (V_{\text{old}}^* + nL) - nL \\ &= \frac{3}{2} V_{\text{old}}^* + \frac{1}{2} nL, \end{aligned}$$

So get factor of $\frac{3}{2}$ plus a constant factor
depending on longest edge length.

Knapsack problem: (Ibarra & Kim, 1975; Lawler, 1979.)

For a fixed approximation ratio $(1-\epsilon)$:

- Find the "large" items that are candidates for inclusion in the optimal solution.
- Solve for the "large" items in a reduced version of the problem, using dynamic programming.
- Find the largest ratio "small" items that can be packed in the remaining volume of the knapsack.

Total running time: $O(n \log_2(\frac{1}{\epsilon}) + \frac{1}{\epsilon^4})$

Thus, this is a FPAS.

Minimum makespan: (Hoogeboom & Shmiz, 1987)

Given: n jobs with ^{integral} processing time p_j
 m identical machines

Makespan: time at which last job finishes.

Objective: minimize the makespan.

Strongly NP-complete.

A PAS exists for this problem, with runtime exponential in $\frac{1}{\epsilon}$.

There are problems for which it is known no FAS or IAS can exist.

Eg: TSP: can't achieve any ratio.

Eg: MAX3SAT:

Can reduce SAT to MAX3SAT using ϵ and then

$$X \in \text{SAT} \Rightarrow \text{MAX3SAT}(\tau(X)) = k$$

$$X \notin \text{SAT} \Rightarrow \text{MAX3SAT}(\tau(X)) < k/(1+\epsilon)$$

for $\epsilon = 0.027$. ~~It is known that~~

Thus, achieving an approximation ratio of $(1+\epsilon)$ for MAX3SAT is NP-hard, since otherwise we could solve SAT in polynomial time.

An algorithm with ratio $\approx 1/0.931$ is known, but there is a gap between the bounds

Eg: Repeated city TSP:

Can't guarantee within a factor of 2. (so $\epsilon=1$)

Eg: MAXCUT:

Can't guarantee within a factor of 1.012. (so $\epsilon=0.012$)

Getting within ~~one~~ one of optimum

Edge coloring: ~~Graph~~ If two ^{edges} ~~edges~~ share ^{a vertex} ~~an edge~~, give them different colors.

Let $\Delta = \max$ degree of a vertex in the graph.

Vizing showed the minimum number of colors needed is either Δ or $\Delta + 1$.

This has been turned into a poly time ~~approximation~~ algo.

~~Optimal value~~ ~~approximation number~~

Min max-degree spanning tree

Find a spanning tree which ~~the~~ has max degree as small as possible.

Can also guarantee to be within 1 in poly time.

Note: If a graph has a Hamiltonian ~~graph~~ ^{path}, then it has a spanning tree with max degree 2.

So: ~~also use~~ min-max spanning tree algo

- If max degree = 2: done, have a ~~the~~ H.P.
- If max degree ≥ 4 : done. ~~is~~ H.P.
- If max degree = 3: maybe, maybe not.