Load Balancing/Multiprocessor Scheduling

$m$ identical machines $M_1, M_2, \ldots, M_m$

$n$ jobs/tasks/items $J_1, J_2, \ldots, J_n$

job $i$ has size/load/processing time $p_i$

**Goal:** assign jobs to machines to minimize maximum load
Greedy Algorithm / List Scheduling

Order items arbitrarily (list)

for $i = 1$ to $n$ do

assign item $i$ to machine with least current load

update load of machine that receives item $i$
Greedy Algorithm / List Scheduling

Order items arbitrarily (list)
for $i = 1$ to $n$ do
    assign item $i$ to machine with least current load
    update load of machine that receives item $i$

Theorem: List scheduling with any list is a 2-approx
Analysis of list scheduling

\[ \text{OPT} \geq \text{average load} = \frac{\sum_i p_i}{m} = \text{LB1} \]
\[ \text{OPT} \geq \text{max job size} = \max_i p_i = \text{LB2} \]

\( L^i_j \): load on machine \( j \) after \( i \) jobs assigned
\[ L = \max_j L^i_j \]

Lemma: \( L \leq \text{LB1} + \text{LB2} \)

Corollary: If \( p_{\text{max}} \leq \varepsilon \text{ LB1} \) then \( L \leq (1+\varepsilon) \text{ OPT} \)
Proof of Lemma

Let $L = L^k_j$

That is the load reached $L$ when assigning job $k$ to machine $M_j$

Since $M_j$ was the least loaded machine when $k$ was scheduled on it

$L^k_{i-1} \geq L - p_k$ for $1 \leq i \leq m$

that is load on all machines is at least $L - p_k$
Proof of Lemma

that is load on all machines is at least $L - p_k$

If all machines have load at least $L-p_k$ then
\[ (\sum_{i=1}^{n} p_i) - p_k \geq m (L - p_k) \]

which implies
\[ L \leq (\sum_{i=1}^{n} p_i)/m \ + \ (1-1/m) \ p_k \]
\[ \leq LB1 \ + \ (1-1/m) \ LB2 \]
\[ \leq (2-1/m) \ OPT \]
LPT List Scheduling

Order items in non-decreasing size
\[ p_1 \geq p_2 \geq \ldots \geq p_m \]

Homework:
LPT scheduling provides a \( \frac{4}{3} \) approximation

Improved lower bound:
\[ \text{OPT} \geq p_m + p_{m+1} \]
A PTAS for Multi-processor Scheduling

Problem is strongly NP-hard
Show reduction from 3-Partition

Therefore no FPTAS

FPTAS for the case when $m$ is a fixed constant
(homework)

PTAS for arbitrary $m$
PTAS

Guess $\text{OPT}$ (more on guessing later)
Scale processing times so that $\text{OPT} = 1$
and hence $p_{\text{max}} \leq 1$

Item $i$ large if $p_i \geq \varepsilon$, else small

Pack large items with load $\leq 1 + \varepsilon$

Pack small items greedily using list scheduling
Packing large items

Round item sizes using geometrically increasing interval sizes:

Let $\delta > 0$

Rounding:

if $p_i \in (\varepsilon(1+\delta)^j, \varepsilon(1+\delta)^{j+1}]$ set $p'_i = \varepsilon(1+\delta)^{j+1}$

Claim: If items can be packed with load 1 then rounded items can be packed with load $1+\delta$
Packing large items

Rounding:
if \( p_i \in (\varepsilon(1+\delta)^j, \varepsilon(1+\delta)^{j+1}] \) set \( p_i' = \varepsilon(1+\delta)^{j+1} \)

Claim: If items can be packed with load 1 then rounded items can be packed with load \( 1+\delta \)
Choose \( \delta = \varepsilon \)

Claim: Number of distinct large item sizes after rounding is \( O(\log (1/\varepsilon) / \varepsilon) \)
Packing large items

Lemma: Instance with $k$ distinct sizes can be solved exactly in time $O(n^{2k})$ time.

Proof: dynamic programming.

Corollary: Large items can be packed (if possible) with load $1+\varepsilon$ in time $O(n^{4\log (1/\varepsilon) /\varepsilon})$. 
Packing small items

Lemma: Suppose large items are packed with load $L$. Then greedy list scheduling of small items results in a load $L'$ s.t

$$L' \leq \max(L, LB1 + \varepsilon) \leq (1+\varepsilon) \text{OPT} \text{ if } L \leq (1+\varepsilon) \text{OPT}$$

Proof: exercise
Guess $\text{OPT} \geq \max (\text{LB1}, \text{LB2})$

Classify jobs as large and small
Round large job sizes
Check if rounded large jobs can be scheduled with load $(1+\varepsilon) \text{OPT}$ using dynamic programming
If large jobs cannot be packed then guess is incorrect
Schedule small jobs on top of large jobs
Relaxed decision procedure

More formally the algorithm is a relaxed decision procedure:

Given a guess $g$ for $OPT$ the algorithm will do one of the following

1. output a schedule with load $(1+\varepsilon)g$ in time $n^{O(\log (1/\varepsilon)/\varepsilon)}$
2. output correctly that there is no schedule of load $g$

One can use above with binary search over $g$ to compute a $(1+\varepsilon)OPT$ approximation
Binary search to implement guessing

Lower bound on $OPT = LB = \max\{LB_1, LB_2\}$
Upper bound on $OPT = UB = 2 \max\{LB_1, LB_2\}$
(from list scheduling analysis)
do binary search on interval $[LB, UB]$ until interval length shrinks to $\varepsilon LB$

Takes $O(\log 1/\varepsilon)$ searches
Total running time is $n^{O(\log (1/\varepsilon) / \varepsilon)}$
Bin Packing

\[ n \text{ items with sizes } s_1, s_2, \ldots, s_n \in (0, 1] \]

bins of capacity 1

Goal: find minimum number of bins in which all items can be packed
Greedy approaches for bin packing

Order items in some way
for $i = 1$ to $n$ do
    if item $i$ can be packed in some *open* bin, pack it
else
    *open* a new bin and pack $i$ in the new bin

Greedy: *open* a new bin only if necessary
Various rules for picking bin

What if several open bins can fit item \( i \)?

FirstFit: pack item in the *first (earliest opened)* bin
LastFit: pack item in the *last* bin to be opened
BestFit: least amount of space left
WorstFit: most amount of space left

NextFit:
Greedy is a 2-approx

Any greedy rule yields a 2-approximation

\[ \text{OPT} \geq \sum_i s_i \]
Greedy is a 2-approx

Any greedy rule yields a 2-approximation

$$\text{OPT} \geq \sum_i s_i$$

A bin is $\alpha$-full if items occupy space atmost $\alpha$

Claim: Greedy has at most one bin that is $1/2$-full 
(why?)
Greedy is a 2-approx

OPT ≥ ∑ᵢ sᵢ

Claim: Greedy has at most one bin that is 1/2-full

let m be the number of bins opened by Greedy
from claim, ∑ᵢ sᵢ > (m-1)/2

OPT > (m-1)/2

⇒ m < 2OPT + 1
⇒ m ≤ 2OPT
Improving Greedy

Order items in non-increasing order
\[ s_1 \geq s_2 \geq \ldots \geq s_n \]

FirstFit is known to give \( \frac{11}{9} \text{OPT} + c \) bins where \( c \) is some fixed constant

Can you prove a bound of \( \frac{3}{2} \text{OPT} + 1 \) bins for FirstFit?
Can Bin Packing have a PTAS?

Partition:

\( n \) items with sizes \( s_1, s_2, \ldots, s_n \)

Q: Can items be partitioned into two sets \( A, B \) such that \( s(A) = s(B) = \sum_i s_i / 2 \)?

Claim: If Bin Packing has a \( 3/2-\varepsilon \) approximation for any \( \varepsilon > 0 \), can solve Partition
Partition via Bin Packing

Create a bin packing instance from a Partition instance

Given items in Partition instance $s_1, s_2, \ldots, s_n$
create items in bin packing instance
$s'_1, s'_2, \ldots, s'_n$ where $s'_i = s_i / x$
and $x = \sum_i s_i / 2$

If Partition is a yes instance, Bin Packing instance has a
solution of size 2
otherwise solution requires at least 3 bins
Asymptotic PTAS

Note that FirstFit obtains a bound of $\frac{11}{9} \text{OPT} + c$
So can beat $3/2$ if we allow an additive constant!

Asymptotic approximation ratio:
for minimization problems
$\text{Alg}(I) \leq \alpha \text{OPT}(I) + \beta$

where $\beta$ is an absolute constant independent of the input size
$\alpha$: asymptotic approximation ratio