Lecture 6: A PTAS for Minimum Makespan

Vazirani (Chapter 10)

In this lecture we will derive algorithms A_ϵ for solving the *Minimum Makespan* problem. A_ϵ

- Will return a schedule with makespan $\leq (1 + 3\epsilon)\, OPT$
- Will run in time $O\left(n^{2k} \lceil \log_2 \frac{1}{\epsilon} \rceil\right)$ where $k = \lceil \log_{1+\epsilon} \frac{1}{\epsilon} \rceil$

A_ϵ is therefore a *Polynomial Time Approximation Scheme (PTAS)* but not a *Fully* Polynomial Time Approximation Scheme (FPAS). (In a FPAS running time is not only polynomial in n but also in $\frac{1}{\epsilon}$.)

Recall the definitions from Lecture 2:

Given processing times for n jobs, $p_1, p_2, \ldots, p_n$, and an integer m, the *Minimum Makespan Scheduling* problem is to find an assignment of the jobs to m machines, $M_1, M_2, \ldots, M_m$ so that the final completion time (the makespan) is minimized. This is an NP-Hard problem.

We already saw a 2-approximation algorithm for this problem. In our analysis of the algorithm we actually proved that

$$LB \leq OPT \leq 2LB$$

where $LB = \max\left\{\frac{1}{m}\Sigma_i\, p_i,\ \max_i\{p_i\}\right\}$.

Makespan is *Dual* to Bin Packing

There is a nice relationship between Makespan and Bin-Packing:

There exists a schedule with makespan t
if and only if
n objects of sizes $p_1, p_2, \ldots, p_n$, can be packed into m bins of capacity t.

Let I be the set of sizes $p_1, p_2, \ldots, p_n$, and $bins(I, t)$ the minimum number of bins needed to pack these objects. Then

$$OPT(\text{makespan}) = \min\{t \,:\, bins(I, t) \le m\}.$$

We know that the t for which this equation minimizes satisfies

$$LB \le t \le 2LB.$$

The basic algorithmic idea will be to binary search $[LB, 2LB]$ to find the minimum t for which $bins(I, t) \le m$.
Note that we can't do this exactly!

The PTAS is based on a *core algorithm* that in $O(n^{2k})$ time finds a bin packing of I that uses $\alpha(I,t,\epsilon)$ bins of size $t(1+\epsilon)$. This packing has the property that

$$\alpha(I,t,\epsilon) \leq bins(I,t).$$

Thus $\forall\epsilon$, $\alpha(I,2LB,\epsilon) \leq bins(I,2LB) \leq m$.
We can now describe the PTAS:

- If $\alpha(I,LB,\epsilon) \leq m$ then use packing given by core algorithm for $t = LB$. This has makespan

  $$\leq LB(1+\epsilon) \leq (1+\epsilon)OPT.$$

- If $\alpha(I,LB,\epsilon) > m$, perform a binary search to find an interval $[T',T] \subseteq [LB,2LB]$ with $T - T' \leq \epsilon LB$, $\alpha(I,T',\epsilon) > m$ and $\alpha(I,T,\epsilon) \leq m$. Return the packing given by the core algorithm with $t = T$.

  Notice that $m < \alpha(I,T',\epsilon) \leq bins(I,T')$ so $T' \leq OPT$ and

  $$\begin{aligned} T &\leq T' + \epsilon LB \\ &\leq Opt + \epsilon Opt = (1+\epsilon)OPT \end{aligned}$$

  Since the core algorithm for $T = t$ returns a schedule with makespan at most $(1+\epsilon)T$ the makespan of the schedule returned is at most

  $$(1+\epsilon)T \leq (1+\epsilon)^2 OPT \leq (1+3\epsilon)OPT.$$

To finish, note that the binary search uses at most $\lceil \log_2 \frac{1}{\epsilon} \rceil$ steps so the running time is as claimed.

Exact Restricted Binpacking

Suppose we are given n items to pack into bins of size t but know that they can only have k different possible sizes. The input to the problem is $I = (i_1, i_2, \ldots, i_k)$ where i_j is the number of items of size j. Let $BINS(i_1, i_2, \ldots, i_k)$ be the minimum number of bins needed to pack these items.
Now suppose we are given input $(n_1, n_2, \ldots, n_k)$, $\Sigma_i\, n_i = n$.

We first compute $\mathcal{Q}$, the set of all k-tuples $(q_1, q_2, \ldots, q_k)$, such that $BINS(q_1, q_2, \ldots, q_k) = 1$. There are at most $O(n^k)$ such tuples and they can be found in $O(n^k)$ time.
We now use *dynamic programming* to fill in *all* entries of the table $BINS(i_1, i_2, \ldots, i_k)$ where $0 \le i_j \le n_j$.

1. $\forall q \in \mathcal{Q}$ set $BINS(q) = 1$.
2. If $\exists j$, such that $i_j < 0$ then $BINS(i_1, i_2, \ldots, i_k) = \infty$.
3. For all other q, use recurrence relation

$$BINS(i_1, i_2, \ldots, i_k) = 1 + \min_{(q_1, q_2, \ldots, q_k) \in \mathcal{Q}} BINS(i_1 - q_1, i_2 - q_2, \ldots, i_k - q_k).$$

When used in the proper order (which?) this recurrence relation calculates all of the entries. Since there $O(n^k)$ entries and each one takes $O(n^k)$ time to calculate, this algorithm requires $O(n^{2k})$ time.

The Core Algorithm for (I, t, ϵ)

Note that $t \in [LB, 2LB]$ so $\forall j$, $p_j \le t$.

1. An object in I is *small* if it has size $\le \epsilon t$.
2. Non-small objects are *rounded* as follows.
   If $p_j \in [t\epsilon(1+\epsilon)^i, t\epsilon(1+\epsilon)^{i+1}]$ then set $p'_j = t\epsilon(1+\epsilon)^i$.
   There can be at most $k = \lceil \log_{1+\epsilon} \frac{1}{\epsilon} \rceil$ different sizes.
3. Use dynamic programming algorithm to optimally pack all non-small objects using p'_j costs into bins of size t.

   Note that rounding can reduce by a factor of $1+\epsilon$ so packing created is valid for $(1+\epsilon)t$ bins with the original p_j sizes.
4. Now place the small items into the $((1+\epsilon)t)$ packing using the greedy algorithm. That is, place them greedily in unused spaces in the bins. Open new bins only if needed. Note that if new bins are opened then all other bins must be filled to height at least t.
5. Let $\alpha(I, t, \epsilon)$ be number of bins used. Recall that these bins are of size $(1+\epsilon)t$.

Lemma: $\alpha(I, t, \epsilon) \leq bins(I, t)$.
Proof:
If the algorithm does open new bins then all of the bins, except possibly the last one, are filled to at least size t. Thus, the optimal packing into bins of size t must use at least $\alpha(I, t, \epsilon)$ bins.

If the algorithm does not open any new bins let I' be the set of non-small items. Then

$$\begin{aligned} \alpha(I, t, \epsilon) &= \alpha(I', t, \epsilon) \\ &\leq bins(I', t) \\ &\leq bins(I, t) \end{aligned}$$

The optimal packing of I' uses $bins(I', t)$ bins. Thus the same packing of the rounded down I' also uses $bins(I', t)$ bins. But $\alpha(I', t, \epsilon)$ is the *optimal* number of bins needed for the rounded down I'. The first inequality follows. The second inequality comes from the fact that optimally packing more items can not reduce the number of bins used.