Approximation Algorithms Recap

An algorithm \( A \) is an \( \alpha \)-approximation for problem \( P \) if,
- \( A \) runs in polynomial time
- \( A \) always outputs a solution with value \( s \) within an \( \alpha \) factor of \( \text{Opt} \)

- Here \( P \) is an optimisation problem with optimal solution of value \( \text{Opt} \)
- If \( P \) is a maximisation problem, \( \frac{\text{Opt}}{\alpha} \leq s \leq \text{Opt} \)
- If \( P \) is a minimisation problem, \( \text{Opt} \leq s \leq \alpha \cdot \text{Opt} \)

We have seen:
- a 2-approximation for \( \text{MAXSAT} \)
- a 3/2-approximation for Bin Packing
  (and a faster 2-approximation)

Scheduling Jobs on Parallel Machines

1. \( m \) identical machines
2. \( n \) jobs
3. \( \text{Goal: minimise the (wall-clock) time taken to process all jobs} \)
   (it’s \( \text{NP-hard} \))
4. time taken

Algorithm: Put job \( j \) on the machine \( i \) with smallest (current) load

\[ O(nm) \] time naively, \( O(n \log m) \) time using a priority queue
(it’s also an online solution)

How good is it?

The greedy approximation

Let \( \text{Opt} \) denote the time taken by the optimal scheduling of jobs
Let \( T_g \) denote the time taken by the greedy schedule

\[ L_i \text{ is the load of machine } i \]

Theorem The greedy algorithm given is a 2-approximation

- Before we prove this, we prove two useful facts,
- \[ \text{Fact } \text{Opt} \geq \max_j t_j \]
  - Some machine, must process the largest job

- \[ \text{Fact } \text{Opt} \geq \frac{\sum_j t_j}{m} \]
  - There is a total of \( \sum_j t_j \) time units of work to be done
  - Some machine \( i \) must have load \( L_i \) at least \( \frac{\sum_j t_j}{m} \)
    (the machines can’t all have below average load)
Proof

Before we prove this, we prove another useful fact and a Lemma

Fact

Let \( g_i \) denote the time taken by the greedy schedule on the machine with largest load

-machine with smallest (current) load was assigned at least two jobs

Then we are done so assume \( m \) with smallest (current) load

\( i \) is the jobs \( \sum_j^m L_j \)

So . . .

\( L_i - t_j \leq L_k \) for all \( 1 \leq k \leq m \)

If we then sum over all \( k \), \( m(L_i - t_j) \leq \sum_{k=1}^m L_k \)

so \( L_i - t_j \leq \sum_{j=1}^m L_j \leq \text{Opt} \) (by the second fact)

Also \( t_j \leq \text{Opt} \) (by the first fact)

Therefore, \( T_g = L_i = (L_i - t_j) + t_j \leq \text{Opt} + \text{Opt} = 2 \text{Opt} \)

Theorem The LPT algorithm is a \( 3/2 \)-approximation

Proof Consider the machine \( i \) with largest load \( T_i = L_i \)

-Let \( j \) denote the last job machine \( i \) completes

-When job \( j \) was assigned, machine \( i \) had the smallest load, \( L_i - t_j \)

\( L_i - t_j \leq L_k \) for all \( 1 \leq k \leq m \)

If there are at most \( n > m \) then \( \text{Opt} \geq 2t_{(m+1)} \) (after sorting)

Job \( j \) takes \( t_j \) time units

Lemma If \( n > m \) then \( \text{Opt} \geq 2t_{(m+1)} \) (after sorting)

Using the same argument as before, we have that,

\( (L_i - t_j) \leq \text{Opt} \)

If \( n \leq m \) then we are done so assume \( n > m \)

Further if \( (L_i - t_j) = 0 \) then \( T_i = L_i = t_j \leq \text{Opt} \)

so assume that \( (L_i - t_j) > 0 \)

Therefore \( i \) was assigned at least two jobs

By the algorithm description, we have that \( j \geq m + 1 \)

\( t_j \leq t_{m+1} \leq \text{Opt}/2 \) (by the Lemma)

Therefore, \( T_g = L_i = (L_i - t_j) + t_j \leq \text{Opt} + \text{Opt}/2 = (3/2) \cdot \text{Opt} \)
Scheduling conclusions

Theorem The greedy algorithm is a 2-approximation which runs in \( O(n \log n) \) time and it’s online

Theorem The LPT algorithm is a \( \frac{3}{2} \) -approximation which runs in \( O(n \log n) \) time

- In fact, LPT is a \( \frac{4}{3} \) -approximation (using better analysis)

\( k \)-centers

\( n \) points (sites) in 2D space

Select \( k \) sites to be centers

The distance between points \( s_i, s_j \) is \( \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \)

(i.e. ‘normal’ euclidean distance)

Goal Minimise the largest distance from any site to the closest center

\( k \)-centers

\( n \) points (sites) in 2D space

Select \( k \) sites to be centers

The distance between points \( s_i, s_j \) is \( \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \)

Goal Minimise the largest distance from any site to the closest center

\( k \)-centers

\( n \) points (sites) in 2D space

Select \( k \) sites to be centers

The distance between points \( s_i, s_j \) is \( \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \)

Goal Minimise the largest distance from any site to the closest center

(\( r \) is in general it’s \( NP \)-hard)

\( k \)-centers

\( n \) points (sites) in 2D space

Select \( k \) sites to be centers

The distance between points \( s_i, s_j \) is \( \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \)

Goal Minimise the largest distance from any site to the closest center
A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center

A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center

A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center

A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center

A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center

A Greedy approximation
Start by picking any point to be a center
Repeatedly pick the site which is furthest from any existing center
A Greedy approximation

The Greedy approximation

Theorem The Greedy algorithm for \( k \)-center is a 2-approximation

Proof

- Let \( C_g/C_{opt} \) denote the set of centers selected by Greedy/Optimal
- Let \( r_g/\text{Opt} \) denote largest site-center distance using Greedy/Optimal

Case 1: No \( s_i, s_i' \in C_g \) are closest to the same \( s_j \in C_{opt} \)

Optimal centers

Distance at most 2Opt so \( r_g \leq 2Opt \)

Greedy centers

Site

Disclaimer: for illustrative purposes only

Case 2: Some \( s_i, s_i' \in C_g \) are closest to the same \( s_j \in C_{opt} \)

- Assume wlog. that Greedy made \( s_i \) a center after \( s_i' \)
- \( s_i \) was added as a center because it was the furthest from any existing Greedy center
- However, \( s_i \) is at most 2Opt away from \( s_i' \)
- So even before adding \( s_i \) as a center, all sites were \( \leq 2\text{Opt} \) away from a Greedy center

Therefore, \( r_g \leq 2\text{Opt} \)

\( k \)-center Conclusions

Theorem The Greedy algorithm for \( k \)-center is a 2-approximation which runs in \( O(nk) \) time

- The approximation works for any (metric) distance function, \( d(s_i, s_j) = L_1, L_{\infty} \) for example
- Distance function \( d \) is a metric iff
  \[
  d(x, y) = d(y, x), d(x, y) \geq 0 \]
  \[
  (d(x, y) = 0 \iff x = y) \text{ and } d(x, z) \leq d(x, y) + d(y, z)
  \]
- For a general (metric) \( d \), the problem is not \( \alpha \)-approximable with \( \alpha < 2 \)
- For \( d = L_2 \), the problem is not \( \alpha \)-approximable with \( \alpha < \sqrt{3} \approx 1.73 \)
- For \( d = L_1 \) or \( d = L_{\infty} \), the problem is not \( \alpha \)-approximable with \( \alpha < 2 \)