Chapter 2:
Greedy Algorithms and Local Search

(cp. Williamson & Shmoys, Chapter 2)

Scheduling Jobs with Due Dates on a Single Machine

Given: n jobs $j = 1, \ldots, n$ with processing time $p_j \geq 0$, release date $r_j \geq 0$ and due dates d_j, $j = 1, \ldots, n$.

Task: Schedule each job nonpreemptively for p_j units of time, starting no earlier than time r_j, such that no two jobs overlap.

Objective: Minimize the maximum lateness $L_{\max} := \max_{j=1,\ldots,n} L_j$ with $L_j := C_j - d_j$ where C_j denotes the completion time of job j, $j = 1, \ldots, n$.

Theorem 2.1.

Deciding whether $L^*_{\max} \leq 0$ is strongly *NP*-complete.

Proof: Polynomial transformation of the 3-Partition Problem. □

Corollary 2.2.

There is no α-approximation algorithm for the scheduling problem for any α, unless $P = NP$.

Proof:... □

Greedy 2-Approximation Algorithm for Negative Due Dates

For a subset of jobs $S \subseteq \{1, \ldots, n\}$ let:

$$r(S) := \min_{j \in S} r_j \qquad p(S) := \sum_{j \in S} p_j \qquad d(S) := \max_{j \in S} d_j$$

Lemma 2.3.

Let $L^*_{\max}$ denote the optimal value. For each subset S of jobs

$$L^*_{\max} \geq r(S) + p(S) - d(S) \ .$$

Proof:... □

Algorithm EDD (earliest due date): Whenever the machine is idle, start to process among all available jobs the one with the earliest due date.

Theorem 2.4.

For the case of non-positive due dates $d_j \leq 0$ for all jobs j, Algorithm EDD is a 2-approximation algorithm.

Proof:... □

The k-Center Problem

Given: A finite metric space V with distances d_{ij} for $i, j \in V$ and $k \in \mathbb{N}$.

Task: Find k centers in V, i. e., $S \subseteq V$ with $|S| = k$.

Objective: Minimize $\max_{i \in V} d(i, S)$ where $d(i, S) := \min_{j \in S} d_{ij}$.

Greedy Algorithm:

1. pick arbitrary $i \in V$ and set $S := \{i\}$;
2. while $|S| < k$ let $j := \arg\max_{j \in V} d(j, S)$ and set $S := S \cup \{j\}$;

Theorem 2.5.

The algorithm is a 2-approximation algorithm for the k-Center Problem.

Proof:... □

Theorem 2.6.

There is no α-approximation algorithm for the k-center problem for $\alpha < 2$, unless $P = NP$.

Proof: Reduction from Dominating Set Problem... □

Scheduling Jobs on Identical Parallel Machines

Given: n jobs $j = 1, \ldots, n$ with processing time $p_j \geq 0$, $j = 1, \ldots, n$, and m identical parallel machines.

Task: Process each job j nonpreemptively for p_j units of time on one of the m machines. A machine can process at most one job at a time.

Objective: Minimize the maximum machine load, i. e., the maximum completion time $C_{\max} := \max_{j=1,\ldots,n} C_j$ (makespan).

Theorem 2.7.

This scheduling problem is strongly *NP*-hard.

Proof: Reduction from 3-Partition. . . □

Local Search

Local Search Algorithm

1. start with an arbitrary schedule;
2. let $j := \arg\max_{j=1,\ldots,n} C_j$;
3. if there is a machine i with load $< C_j - p_j$, reassign j to i and goto 2;

Theorem 2.8.

When the algorithm terminates, the makespan of the final solution is at most twice the optimum makespan.

Proof:. . . □

Lemma 2.9.

If the Local Search Algorithm always moves job j to a currently least loaded machine, it terminates after at most n iterations.

Proof:. . . □

List Scheduling

List Scheduling Algorithm

1. start with the empty schedule and consider the jobs in arbitrary order;
2. always assign the next job to the currently least loaded machine;

Theorem 2.10.
The List Scheduling Algorithm is a 2-approximation algorithm.

Proof:. . . □

The variant of list scheduling where jobs are considered in non-increasing order is called the longest processing time (LPT) rule.

Theorem 2.11.
The LPT rule is a 4/3-approximation algorithm.

Proof:. . . □

Later: There is a PTAS for this scheduling problem!

Traveling Salesperson Problem (TSP)

Theorem 2.12.
There is no α-approximation algorithm for the TSP for any α (e. g., $\alpha = 2^n$), unless $P = NP$.

Proof: Reduction from Hamiltonian Circuit. . . □

In the following we thus consider the metric TSP where distances between cities fulfill the triangle inequalities.

TSP: Nearest Insertion Algorithm

Nearest Insertion Algorithm

1. start with tour through two cities $S := \{i, j\}$ of minimum distance d_{ij};
2. for each uninserted city k, compute the minimum distance $d(k, S)$ between k and a city in the current tour;
3. let $\ell := \arg\min_{k \notin S} d(k, S)$ and add ℓ to the tour after its nearest city;
4. set $S := S \cup \{\ell\}$; if $|S| < n$, then goto 2;

Theorem 2.13.

The Nearest Insertion Algorithm is a 2-approximation algorithm for the metric TSP.

Proof:. . . □

The Approximability of the Metric TSP

Remarks:

- The Nearest Insertion Algorithm is closely related to Prim's Algorithm and to the double-tree 2-approximation algorithm for the metric TSP.
- Christofides' Algorithm is a 3/2-approximation algorithm for the metric TSP (see ADM II).

Theorem 2.14 (Papdimitriou & Vempala 2006).

There is no α-approximation algorithm for the metric TSP for $\alpha < 220/219$, unless $P = NP$.

- Notice, however, that there is a PTAS for the Euclidean TSP which is a special case of the metric TSP!

Minimum-Degree Spanning Trees

Given: Graph $G = (V, E)$.

Task: Find spanning tree T of G minimizing maximum node degree $\Delta(T)$.

Theorem 2.15.

It is *NP*-complete to decide whether or not a given graph has a spanning tree of maximum degree two.

Proof: Reduction of Hamiltonian Path. □

Local Search for Minimum-Degree Spanning Tree

Local improvement step:

- consider some spanning tree T and a node u of degree $d_T(u)$ in T;
- look for an edge $e = vw$ not in T such that
  - adding e to T closes a cycle C containing node u;
  - $\max\{d_T(v), d_T(w)\} \leq d_T(u) - 2$;
- add e to T and remove a tree-edge incident to u destroying C;

Notice: The local improvement step reduces the degree of u by one and only increases the degrees of v and w by at most one.

Local Search Algorithm

1. start with an arbitrary spanning tree T;
2. iteratively do a local improvement step for some node u with
$$d_T(u) \geq \Delta(T) - \lceil \log_2 n \rceil \ ;$$

The algorithm terminates if a local improvement step for a node u with $d_T(u) \geq \Delta(T) - \lceil \log_2 n \rceil$ is no longer possible. Then, T is locally optimal.

Analysis

Theorem 2.16.

Let T be a locally optimal tree. Then $\Delta(T) \leq 2\,\mathrm{OPT} + \lceil \log_2 n \rceil$.

Proof:... □

Theorem 2.17.

The Local Search Algorithm fins a locally optimal tree in polynomial time.

Proof:... □

We finally state the following result without proof:

Theorem 2.18.

There is a polynomial-time algorithm which finds a spanning tree T with $\Delta(T) \leq \mathrm{OPT} + 1$.