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 $\Delta(T) \leq \mathrm{OPT} + 1$.