Differential Geometry

Instructor: Jianfeng Lin

Notes Taker: Zejin Lin

TSINGHUA UNIVERSITY.

linzj23@mails.tsinghua.edu.cn

lzjmaths.github.io

February 18, 2025

Contents

1 Introduction

Discrete (combinatorial) optimization is a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solution is discrete or can be reduced to a discrete set.

However, usually this feasible solution set is very large (due to combinatorial explosion) and it is computationally infeasible to go through all feasible solutions and find the one with opimal objective function value.

Example 1.1 (Task Assignment)**.** There are n tasks and n workers. Each task has an importance score a_i and each worker has a skill level b_i. We need to assign each task to a worker such that the sum of $\sum_{i=1}^{n} a_i b_{\sigma(i)}$ is maximized.

1.1 Models of Computation: Turing Machines

Definition 1.2 (A Deterministic Turing Machine(DTM))**.** It consists of an infinitely-long tape (memory) and a deterministic finite automata that controls the head to move along the tape and read/write symbols from/to the tape cells.

Definition 1.3 (Complexity measure)**.** Running time is the number of steps of Turing machine.

Memory is the number of tape cells used.

Definition 1.4 (Caveat)**.** No random access of memory

- Single-tape DTM requires $\geqslant n^2$ steps to detect n bit palindromes.
- EAsy to detect palindromes within c_n steps on a real computer.

1.2 Models of Computation: word RAM

Definition 1.5. Each memory location and input/output cell stores a w-bit integer (assume $w \geqslant \log_2 \omega$).

Primitive Operations:

1.3 Polynomial Running Time

Definition 1.6. We say that an algorithm is **efficient** if its running time is polynomial of input size n.

Example 1.7 (Task machine)**.** Polynomial-time algorithm: selection sort/inserting sort/quick sort/merge sort.

Non-polynomial-time algorithm: try all possible matching and output the one with the highest score.

- Definition is relatively insensitive to model of computation.
- The poly-times algorithm that people develop have both small constants and small exponents
- Breaking through the exponential barrier is a major challenge.

1.4 Notation

Definition 1.8. $f(n)$ is $O(g(n))$ if there exist constants $c > 0$ and $n_0 \geqslant 1$ such that $0 \leqslant f(n) \leqslant c \cdot g(n)$ for all $n \geqslant n_0$.

$f(n)$ is $\Omega(g(n))$ is $g(n) \in O(f(n))$.

$f(n)$ is $\Theta(g(n))$ is both $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$.

1.5 Tentative Syllabus

We will introduce three exact discrete optimization algorithms(6 weeks):

- Greedy algorithms
- Dynamic programming
- Network flows

And some approximation algorithms for intractable discrete optimization problems(9 weeks)

- Definition of approximation algorithms
  - Algorithm techniques: greedy, linear programming relaxation, semidefinite programming relaxation.
- Hardness of approximation
  - Techniques: hardness reductions, Fourier analysis of Boolean functions.
- Problems studied: Set-Cover, facility location, K-center, Multi-Cut, Max-Cut, $\cdots$

2 Greedy Algorithms

Example 2.1 (Interval Scheduling)**.** Input: n jobs, $\{(s_i, f_i)\}_{i=1}^n$. Goal: How to choose jobs with maximized number such that each pair of intervals do not intersect.

Greedy Framework Consider jobs in order $\pi(1), \pi(2), \cdots, \pi(n)$. For each $\pi(i)$, $i = 1, 2, \cdots, n$, if $\pi(i)$ compatible with all selected jobs, then select $\pi(i)$.

The choice of π: Earliest-start-time-first, Earliest-finish-time-first, Longest-job-first, Shortest-job-first, etc.

Theorem 2.2. *Earliest-finish-time-first greedy returns an optimal solution.*

Proof. Suppose algorithm selects $i_1, i_2, \cdots, i_k$, opt selects $k' > k$ jobs.

Choose an optimal solution agrees with algorithm in first r jobs so that r maxmized, $j_1, j_2, \cdots, j_{k'}$.

Obviously, $r < k$. Then $f_{i_{r+1}} < f_{j_{r+1}}$. Therefore, we can replace i_{r+1} with j_{r+1} to get another optimal solution, which contradicts to the fact that r maxmized. □

Example 2.3 (Interval Partitioning)**.** Input: n lectures, $\{(s_i, f_i)\}_{i=1}^n$.

Goal: Position lectures into minimum number of classrooms so that in each classroom lectures are compatible.

Greedy Framework Lectures in order $\pi(1), \cdots, \pi(n)$, the number of opening classrooms is zero in the beginning. For each $\pi(i)$,

If $\exists$ opening classroom j *s.t.* lecture $\pi(i)$ compatible with lectures in j, then $\pi(i) \rightarrow$ classroom j.

Else, open a new classroom for $\pi(i)$.

Proof. Introduce a concept **Depth**: $d(t) =$ Number of lectures active at time t, and $d = \max_t \{d(t)\}$.

Claim. opt $\geqslant d$.

Lemma 2.4. *alg* $\leqslant d$

□

List of Theorems