Lecture Notes

Probabilistic Analysis of Algorithms

Prof. Dr. Heiko Röglin
Department of Computer Science

August 1, 2016

Contents

Please send comments and corrections to roeglin@cs.uni-bonn.de.

Chapter 1

Introduction

The theory of algorithms has traditionally focused on worst-case analysis. This focus has led to both a deep theory and many beautiful and useful algorithms. There are, however, a number of important problems and algorithms for which worst-case analysis does not provide useful or empirically accurate results. One prominent example is the simplex method for linear programming whose worst-case running time is exponential while in fact it runs in near-linear time on almost all inputs of interest. Another example is the knapsack problem. While this problem is NP-hard, it is a very easy optimization problem in practice and even very large instances with millions of items can be solved efficiently. The reason for this discrepancy between worst-case analysis and empirical observations is that for many algorithms worst-case instances have an artificial structure and hardly ever occur in practical applications.

In recent years a paradigm shift towards a more realistic and robust algorithmic theory has been initiated. The development of a more realistic theory hinges on finding models that measure the performance of an algorithm not only by its worst-case behavior but rather by its behavior on typical inputs. However, for an optimization problem at hand it is usually impossible to rigorously define the notion of "typical input" because what such an input looks like depends on the concrete application and on other indistinct parameters. In order to cope with this problem, Spielman and Teng introduced the model of *smoothed analysis* [30]. This model can be considered as a less pessimistic variant of worst-case analysis in which the adversarial input is subject to a small amount of random noise.

Let us illustrate smoothed analysis by means of the knapsack problem. Worst-case analysis can be viewed as a game between an algorithm designer and an adversary. First the algorithm designer chooses an algorithm, then the adversary chooses an input on which the algorithm performs as poorly as possible. For the knapsack problem, the adversary chooses profits, weights, and the capacity of the knapsack. In order to limit the power of the adversary to construct artificial instances that do not resemble typical inputs, we add some randomness to his decisions in the following way. Instead of being able to determine the numbers in the input exactly, he can only choose for each number an interval of length $1/\phi$ from which it is chosen uniformly at random.

Here $\phi \geq 1$ is some parameter that determines the power of the adversary. The larger it is, the more precisely he can specify the input. In the limit for $\phi \to \infty$ the adversary is as powerful as in a worst-case analysis.

We will see in this lecture that there are heuristics for the knapsack problem whose expected running time is polynomially bounded in the number of items and the parameter ϕ. This shows that instances on which the algorithm requires exponential running time are fragile with respect to random perturbations and even a small amount of randomness suffices to rule out such instances with high probability. In practice, random noise can stem from measurement errors, numerical imprecision, or rounding errors. It can also model arbitrary influences that we cannot quantify exactly, but for which there is also no reason to believe that they are adversarial.

After its invention in 2001, smoothed analysis has attracted a great deal of attention and it has been applied in a variety of different contexts, e.g., in multi-objective optimization, local search, clustering, and online algorithms. By now smoothed analysis is widely accepted as a realistic alternative to worst-case analysis. In this lecture we will discuss various results from the area of smoothed analysis.

Chapter 2

Probability Theory

In this chapter, we introduce some notation and review some facts from probability theory. However, this chapter is by no means intended to give a formal introduction to probability theory. Readers who are unfamiliar with probability theory are referred to the excellent books by Mitzenmacher and Upfal [24] and by Motwani and Raghavan [25] for computer science-related introductions. Readers who are familiar with probability theory can skip the sections 2.1 to 2.4. In Section 2.5 we introduce the notion of *ϕ-perturbed numbers* and discuss the probabilistic input model that we will employ throughout this lecture.

We define $\mathbb{N} = \{1, 2, 3, \ldots\}$ to be the set of natural numbers and $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$ to be the set of natural numbers with zero. For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. We use $\mathbb{R}_{\geq 0}$ to denote the set $\{x \in \mathbb{R} \mid x \geq 0\}$. Given a vector $x \in \mathbb{R}^n$, we use $x_1, \ldots, x_n$ to denote its entries, i.e., $x = (x_1, \ldots, x_n)^\mathsf{T}$. The norm $\|x\|$ of a vector $x \in \mathbb{R}^n$ is always meant to be its Euclidean norm, i.e., $\|x\| = \sqrt{x_1^2 + \cdots + x_n^2} = \sqrt{x^\mathsf{T} x}$. We denote the power set of a set M by 2^M.

2.1 Probability Spaces

To formalize the concept of probability, we introduce the notion of a *probability space*, which is used to model random experiments. We start with the definition of a discrete probability space and extend it afterwards to general probability spaces.

Definition 2.1. *A* discrete probability space *is a pair (Ω, p) with the following components.*

1. *The* sample space Ω *is an arbitrary finite or countable infinite set. It represents the set of all possible outcomes of the random experiment modeled by the probability space.*
2. *The* probability measure $p : \Omega \to [0, 1]$ *is a function that satisfies $\sum_{x \in \Omega} p(x) = 1$. It assigns a probability to each outcome of the random experiment.*

Intuitively one can interpret the values of the probability measure as follows: If the random experiment modeled by the probability space (Ω, p) is repeated independently a large number of times then the fraction of trials with outcome $x \in \Omega$ approaches $p(x)$. This is known as the *frequentist interpretation of probability*.

Let us consider some examples.

- If we want to model the random experiment of tossing a fair six-sided die then $\Omega = \{1, 2, \ldots, 6\}$ and $p(x) = 1/6$ for all $x \in \Omega$ are natural choices for the sample space and the probability measure, respectively.
- The random experiment of tossing a fair coin twice can be modeled by the probability space (Ω, p) with $\Omega = \{T, H\}^2$ and $p(x) = 1/4$ for all $x \in \Omega$. Here T and H stand for the tail and head, respectively.
- The random experiment of tossing two fair and indistinguishable coins simultaneously can be modeled by the probability space (Ω, p) with sample space $\Omega = \{\{T, T\}, \{H, H\}, \{T, H\}\}$ and probability measure p with $p(\{T, T\}) = p(\{H, H\}) = 1/4$ and $p(\{T, H\}) = 1/2$.

The elements of the sample space Ω are called *elementary events*. While the probability measure p assigns probabilities only to the elementary events, it is often natural to ask about the probability of a subset of Ω. In the first example above one could, for example, ask about the probability of the die showing an even number. We call every subset of Ω an *event* and extend p to a function $P : 2^\Omega \to \mathbb{R}$ by $P(X) = \sum_{x \in X} p(x)$. Then the probability of the die showing an even number can be computed as $P(\{2, 4, 6\}) = p(2) + p(4) + p(6) = 3/6 = 1/2$.

In this lecture we will often deal with continuous random experiments with uncountable sample spaces. A simple example would be to choose a real number from the interval $[0, 1]$ uniformly at random by which we mean intuitively that each outcome from $\Omega = [0, 1]$ should have the same probability. We cannot model this by a probability measure $p : \Omega \to [0, 1]$ as in the case of discrete probability spaces. Indeed if one wants to assign the same probability to each outcome in $[0, 1]$, then one has no choice but to set $p(x) = 0$ for each $x \in \Omega$. Clearly it makes no sense to extend this function to a function $P : 2^\Omega \to \mathbb{R}$ in the same way as for discrete probability spaces (in particular, because the sum $\sum_{x \in X} p(x)$ is not even defined for uncountable $X \subseteq [0, 1]$). Instead if Ω is uncountable, one has to define the function $P : 2^\Omega \to \mathbb{R}$ directly without recourse to the function p.

Definition 2.2. *A* probability space *is a tuple $(\Omega, \mathcal{F}, P)$ with the following components.*

1. *The* sample space Ω *is an arbitrary set. It represents the set of all possible outcomes of the random experiment modeled by the probability space.*
2. *The* set of events $\mathcal{F} \subseteq 2^\Omega$ *is a* σ-algebra *on Ω. This means that $\mathcal{F}$ is a family of subsets of Ω with the following properties.*
   (a) $\Omega \in \mathcal{F}$
   (b) *$\mathcal{F}$ is closed under complementation:* $X \in \mathcal{F} \implies \Omega \setminus X \in \mathcal{F}$.
   (c) *$\mathcal{F}$ is closed under countable unions:* $X_1, X_2, X_3, \ldots \in \mathcal{F} \implies \bigcup_{i=1}^{\infty} X_i \in \mathcal{F}$.

The set $\mathcal{F}$ represents the events for which a probability is defined.

3. *The* probability measure $P : \mathcal{F} \to [0,1]$ *is a σ-additive function that satisfies $P(\Omega) = 1$. This means that for pairwise disjoint events $X_1, X_2, X_3, \ldots \in \mathcal{F}$ one has $P(\bigcup_{i=1}^{\infty} X_i) = \sum_{i=1}^{\infty} P(X_i)$.*

We will not need to deal very deeply with probability theory in this lecture and we presented the definition above mainly for the sake of completeness. Still let us briefly discuss it. First of all, one might ask what the σ-algebra $\mathcal{F}$ is needed for. In light of Definition 2.1 it seems reasonable to always choose $\mathcal{F} = 2^{\Omega}$ because then a probability is defined for every subset of the sample space. The problem with this is that under the assumption of the continuum hypothesis there is no σ-additive probability measure P on $2^{\mathbb{R}}$ with $P(\{x\}) = 0$ for each $x \in \mathbb{R}$ (Banach-Kurnatowski theorem). Since we need such measures to model, for example, the random experiment in which a real number from the interval $[0,1]$ is chosen uniformly at random, it makes sense to consider other σ-algebras than $2^{\mathbb{R}}$.

We will often consider probability spaces with $\Omega = \mathbb{R}$ in this lecture. Then $\mathcal{F}$ will always be chosen as the *Borel σ-algebra*, which is defined as the smallest σ-algebra containing all intervals (a,b) with $a, b \in \mathbb{R}$. All subsets of $\mathbb{R}$ that occur in this lecture (and in most other contexts) belong to the Borel algebra. This includes, in particular, all closed intervals $[a,b]$, all half-closed intervals $(a,b]$ and $[a,b)$, and the union of countably many such intervals. To define a probability measure $P : \mathcal{F} \to [0,1]$, it suffices in this case to define P only for all open intervals (a,b). Then the value $P(X)$ is uniquely determined for every $X \in \mathcal{F}$ because P is σ-additive. Similarly it suffices to define P only for all closed intervals $[a,b]$.

Let us consider as an example the random experiment that a real number from the interval $[0,1]$ is chosen uniformly at random. This can be modeled by the probability space $(\Omega, \mathcal{F}, P)$ where $\Omega = \mathbb{R}$, $\mathcal{F}$ is the Borel σ-algebra, and $P([a,b]) = b' - a'$ for $[a',b'] := [0,1] \cap [a,b]$ with $b' \geq a'$. Then, in particular, $P(\{x\}) = 0$ and $P([0,x]) = x$ for every $x \in [0,1]$.

A simple property of a probability measure that we will use very frequently in this lecture is the following *union bound*.

Lemma 2.3. *Let $X_1, \ldots, X_n$ be events in some probability space $(\Omega, \mathcal{F}, P)$. Then*

$$P(X_1 \cup \cdots \cup X_n) \leq \sum_{i=1}^{n} P(X_i).$$

Another elementary property of probabilities is the *law of total probability*.

Lemma 2.4. *Let A be an event in some probability space $(\Omega, \mathcal{F}, P)$ and let $B_1, \ldots, B_n$ be events that form a partition of the sample space Ω, that is, $B_i \cap B_j = \emptyset$ for $i \neq j$ and $B_1 \cup \ldots \cup B_n = \Omega$. Then*

$$\mathbf{Pr}[A] = \sum_{i=1}^{n} \mathbf{Pr}[A \cap B_i].$$

2.2 Random Variables

Often one is not interested in the actual outcome of a random experiment but only in one characteristic parameter. Consider the example that we study the performance of some algorithm for the knapsack problem on random inputs with n items in which every profit, every weight, and the capacity are chosen at random. Then the underlying probability space is defined on the set $\Omega = \mathbb{R}^{2n+1}$ because $2n+1$ numbers are chosen at random. However, we do not care about the actual outcome of this random experiment but we are only interested in the running time of the algorithm. In order to model this, we introduce the notion of a *random variable*.

Definition 2.5. *Let $(\Omega, \mathcal{F}, P)$ be a probability space. A (real-valued) random variable $X : \Omega \to \mathbb{R}$ is a function from the set of elementary events to the real numbers that satisfies*

$$X^{-1}\Big([a,b]\Big) := \Big\{z \in \Omega \mid X(z) \in [a,b]\Big\} \in \mathcal{F} \tag{2.1}$$

for every interval $[a,b] \subseteq \mathbb{R}$. A random variable is called discrete *if it takes only values from a countable subset of $\mathbb{R}$. Otherwise it is called* continuous.

In the example discussed before the definition, we can choose as random variable the function X that maps every outcome from $\Omega = \mathbb{R}^{2n+1}$ to the number of steps the algorithm requires on that outcome. Then the probability that the algorithm requires at most T steps on a randomly chosen input can be written as $P(X^{-1}([0,T]))$ where P denotes the probability measure of the underlying probability space on $\Omega = \mathbb{R}^{2n+1}$. Observe that condition (2.1) in Definition 2.5 is only necessary to ensure that for any interval $[a,b]$ the probability of X taking a value in $[a,b]$ is defined by the underlying probability space.

For a random variable X we will denote the probability that it takes some value $x \in \mathbb{R}$ by $\mathbf{Pr}[X = x]$ and we will denote the probability that it takes a value in some interval $[a,b]$ by $\mathbf{Pr}[X \in [a,b]]$. Formally

$$\mathbf{Pr}[X = x] := P(X^{-1}(\{x\})) \qquad \text{and} \qquad \mathbf{Pr}[X \in [a,b]] := P(X^{-1}([a,b])).$$

It is very convenient to describe the behavior of a continuous random variable by a *cumulative distribution function (CDF)* or a *probability density function (PDF)*.

Definition 2.6. *The* cumulative distribution function (CDF) *of a continuous random variable X is the function $F_X : \mathbb{R} \to \mathbb{R}_{\geq 0}$ given by $F_X(t) = \mathbf{Pr}[X \leq t]$ for all $t \in \mathbb{R}$. A function $f_X : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is called* probability density function (PDF) *of X if*

$$\mathbf{Pr}[X \in [a,b]] = \int_a^b f_X(t)\,dt$$

for every interval $[a,b] \subseteq \mathbb{R}$.

Usually we do not mention the underlying probability space of a random variable explicitly, but we are only interested in the distribution or density of the random variable. This is, in particular, true in the following examples.

- Let X be uniformly distributed on $[a, b]$ for some $a < b$. Then the distribution F_X of X is defined by

$$F_X(t) = \mathbf{Pr}[X \leq t] = \begin{cases} 0 & \text{if } t < a, \\ \frac{t-a}{b-a} & \text{if } a \leq t \leq b, \\ 1 & \text{if } t > b, \end{cases}$$

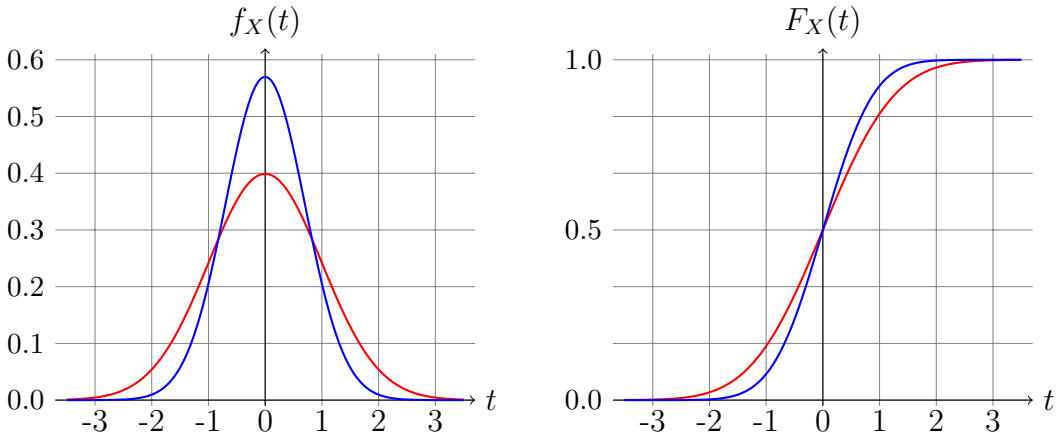
and a density f_X of X is defined by

$$f_X(t) = \begin{cases} 0 & \text{if } t < a \text{ or } t > b, \\ \frac{1}{b-a} & \text{if } a \leq t \leq b. \end{cases}$$

- A distribution that occurs frequently is the *normal distribution* (also known as *Gaussian distribution*). It is used, for example, to model measurement errors in physical experiments. We say that a random variable X is *normally distributed with mean μ and standard deviation σ* if it can be described by the density function f_X with

$$f_X(t) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(t-\mu)^2}{2\sigma^2}}.$$

There does not exist a formula for the distribution of a normal distribution in terms of elementary functions but numerical approximations are known. The following figures show the densities and distributions of two Gaussian random variables with mean 0 and standard deviation 0.7 (blue) and 1 (red).



- The behavior of a discrete random variable X is usually described by its cumulative distribution function or by its *probability mass function*. This function simply gives for each value $t \in \mathbb{R}$ the probability that $X = t$. As an example, let us consider the case that we have a coin that comes up heads with probability $p \in (0, 1]$ and let us assume that we toss this coin until it comes up heads for the first time. Let X denote the total number of coin tosses in this experiment. Then X is a random variable and $\mathbf{Pr}[X = t] = p(1-p)^{t-1}$ for $t \in \mathbb{N}$. This is because X takes the value t if and only if the first $t-1$ tosses come up tails, which happens with probability $(1-p)^{t-1}$, and if the tth toss comes up heads, which happens with probability p. Along the same lines, one can argue that the distribution F_X of X satisfies $F_X(t) = \mathbf{Pr}[X \leq t] = 1 - (1-p)^t$ because X is

larger than t if and only if the first t tosses all come up tails, which happens with probability $(1-p)^t$. The random variable X is called a *geometric random variable with parameter p*.

The following easy property of continuous random variables with bounded density will be used extensively throughout this lecture.

Lemma 2.7. *Let X be a continuous random variable with density function $f_X : \mathbb{R} \to \mathbb{R}_{\geq 0}$ with $f_X(t) \leq \phi$ for all $t \in \mathbb{R}$. Then for every $a \in \mathbb{R}$ and every $\varepsilon \geq 0$, $\mathbf{Pr}[X \in [a, a+\varepsilon]] \leq \varepsilon\phi$.*

Proof. According to Definition 2.6,

$$\mathbf{Pr}[X \in [a, a+\varepsilon]] = \int_a^{a+\varepsilon} f_X(t)\,dt \leq \int_a^{a+\varepsilon} \phi\,dt = \varepsilon\phi. \qquad \square$$

2.3 Expected Values

Often we are not interested in the complete distribution of a random variable but only in some of its key features. We will study, for example, the *expected value* of a random variable, which is the average value, where every outcome is weighted by the probability that it occurs. Similar to the frequentist interpretation of probability, the expected value can be interpreted as the average value of the random variable if the random experiment is repeated independently a large number of times.

Definition 2.8. *Let X be a discrete random variable that takes only values in a countable set $M \subseteq \mathbb{R}$. The* expected value *of X is defined as*

$$\mathbf{E}[X] = \sum_{t \in M} t \cdot \mathbf{Pr}[X = t].$$

The expected value of a continuous random variable X with density f_X is defined as

$$\mathbf{E}[X] = \int_{-\infty}^{\infty} t \cdot f_X(t)\,dt$$

if the integral exists.

We leave the proof of the following lemma as an exercise for the reader.

Lemma 2.9. *For a discrete random variable X that takes only values in $\mathbb{N}_0$, we can write the expected value as*

$$\mathbf{E}[X] = \sum_{t=1}^{\infty} \mathbf{Pr}[X \geq t].$$

For a continuous random variable X that takes only values in $\mathbb{R}_{\geq 0}$, we can write the expected value as

$$\mathbf{E}[X] = \int_{t=0}^{\infty} \mathbf{Pr}[X \geq t]\;dt.$$

Let us consider some examples.

- If X describes the outcome of a die toss, we have $\mathbf{Pr}[X = a] = 1/6$ for every $a \in \{1, 2, \ldots, 6\}$ and $\mathbf{Pr}[X = a] = 0$ for every $a \notin \{1, 2, \ldots, 6\}$. The expected value of X is
$$\mathbf{E}[X] = \sum_{t=1}^{6} \frac{t}{6} = 3.5.$$

- Let X be a geometric random variable with parameter p. Then
$$\mathbf{E}[X] = \sum_{t=1}^{\infty} \mathbf{Pr}[X \geq t] = \sum_{t=1}^{\infty} (1-p)^{t-1} = \sum_{t=0}^{\infty} (1-p)^{t} = \frac{1}{1-(1-p)} = \frac{1}{p}.$$

- Let X be a Gaussian random variable with mean 0 and standard deviation σ. Then the symmetry of the density function around 0 implies that the expected value of X is 0 if it exists. One only needs to argue that the integral exists. This follows from the following calculation:
$$\begin{aligned}
\mathbf{E}[X] &= \int_{-\infty}^{\infty} \frac{t}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2\sigma^2}}\, dt \\
&= \int_{-\infty}^{0} \frac{t}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2\sigma^2}}\, dt + \int_{0}^{\infty} \frac{t}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{t^2}{2\sigma^2}}\, dt \\
&= \frac{1}{\sigma\sqrt{2\pi}} \cdot \Big[-\sigma^2 e^{-\frac{t^2}{2\sigma^2}} \Big]_{-\infty}^{0} + \frac{1}{\sigma\sqrt{2\pi}} \cdot \Big[-\sigma^2 e^{-\frac{t^2}{2\sigma^2}} \Big]_{0}^{\infty} \\
&= \frac{-\sigma^2}{\sigma\sqrt{2\pi}} + \frac{\sigma^2}{\sigma\sqrt{2\pi}} = 0.
\end{aligned}$$

One simple but extremely helpful property of expected values is *linearity of expectation*, as stated in the following lemma.

Lemma 2.10. *Let X and Y be random variables and let $a, b \in \mathbb{R}$. Then*
$$\mathbf{E}[X + Y] = \mathbf{E}[X] + \mathbf{E}[Y] \quad \text{and} \quad \mathbf{E}[aX + b] = a\mathbf{E}[X] + b.$$

To illustrate the usefulness of linearity of expectation, let us consider some examples.

- Let X be a Gaussian random variable with mean μ and standard deviation σ and let Y be a Gaussian random variable with mean 0 and standard deviation σ. Then it is easy to see that X has the same density as $Y + \mu$. Hence,
$$\mathbf{E}[X] = \mathbf{E}[Y + \mu] = \mathbf{E}[Y] + \mu = 0 + \mu = \mu.$$

- Assume that we toss a die k times and let X denote the number of distinct outcomes that occur. For $i \in \{1, \ldots, 6\}$ let $X_i = 1$ if outcome i occurs at least once and $X_i = 0$ otherwise. Then
$$\mathbf{Pr}[X_i = 1] = 1 - \mathbf{Pr}[X_i = 0] = 1 - (5/6)^k$$

and hence,

$$\mathbf{E}[X] = \mathbf{E}\left[\sum_{i=1}^{6} X_i\right] = \sum_{i=1}^{6} \mathbf{E}[X_i] = \sum_{i=1}^{6} (0 \cdot \mathbf{Pr}[X_i = 0] + 1 \cdot \mathbf{Pr}[X_i = 1])$$
$$= \sum_{i=1}^{6} \mathbf{Pr}[X_i = 1] = 6 \cdot (1 - (5/6)^k).$$

Often we are interested in how likely it is that a random variable takes a value that deviates significantly from its expected value. For a non-negative random variable, *Markov's inequality* gives a bound on the probability that it is larger than its expected value by some factor.

Lemma 2.11. *Let X be a random variable that takes only values in $\mathbb{R}_{\geq 0}$ and let $a \geq 1$. Then*

$$\mathbf{Pr}[X \geq a \cdot \mathbf{E}[X]] \leq \frac{1}{a}.$$

2.4 Conditional Probability and Independence

Given two events A and B in a probability space, one is often interested in the probability of event A under the condition that event B occurs.

Definition 2.12. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $A, B \in \mathcal{F}$ be two events with $P(B) > 0$. The* conditional probability of A given B *is defined as*

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}.$$

We consider two examples.

- Let X and Y denote the outcomes of two consecutive tosses of a die and let $Z = X + Y$. We denote by A the event that $Z \geq 10$ and by B the event that $X = 6$. Then $P(A \cap B) = 3/36$ because there are three out of 36 possible outcomes of the two tosses that satisfy $X = 6$ and $Z \geq 10$ (these are the outcomes $(6, 4)$, $(6, 5)$, and $(6, 6)$). Hence,

  $$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{3/36}{1/6} = \frac{1}{2}.$$

  As a comparison the probability of A without conditioning on B is only $1/6$.

- Let X and Y be defined as in the example above. Now let A denote the event that $X = 3$ and let B denote the event that $Y = 4$. Then

  $$P(A \mid B) = \frac{P(A \cap B)}{P(B)} = \frac{1/36}{1/6} = \frac{1}{6}.$$

  In this example the conditional probability of A given B equals the probability of A without conditioning on B.

Intuitively, there is an obvious difference between the two examples discussed above: In the first example, the events A and B are dependent because the probability of A changes if one conditions on B. In the second example, the events A and B are independent in the sense that conditioning of B does not change the probability of A. This is formalized in the following definition.

Definition 2.13. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $A, B \in \mathcal{F}$ be two events. The events A and B are called* independent *if $P(A \cap B) = P(A) \cdot P(B)$. Otherwise they are called* dependent*. Two random variables X and Y are called* independent *if the events $\{X \leq a\}$ and $\{Y \leq b\}$ are independent for any $a, b \in \mathbb{R}$. Otherwise they are called* dependent*.*

It follows easily from the definition that $P(A \mid B) = P(A)$ if and only if A and B are independent. Furthermore one can also prove that $\mathbf{E}[X \cdot Y] = \mathbf{E}[X] \cdot \mathbf{E}[Y]$ for independent random variables X and Y. Let us point out that in contrast to this, Lemma 2.10 (linearity of expectation) is also true for dependent random variables.

For an event A and a random variable X, one can also define a *conditional expectation of X given A*. Intuitively, this is the average value of X given that A occurs.

Definition 2.14. *Let $(\Omega, \mathcal{F}, P)$ be a probability space and let $A \in \mathcal{F}$ be an event with $P(A) > 0$. Let X be a discrete random variable that takes only values in a countable set $M \subset \mathbb{R}$. The* conditional expectation of X given A *is defined as*

$$\mathbf{E}[X \mid A] = \sum_{t \in M} t \cdot \mathbf{Pr}[X = t \mid A].$$

From the previous definition and the law of total probability, one can easily see that

$$\mathbf{E}[X] = \mathbf{Pr}[A] \cdot \mathbf{E}[X \mid A] + \mathbf{Pr}[\neg A] \cdot \mathbf{E}[X \mid \neg A],$$

where $\neg A := \Omega \setminus A$.

2.5 Probabilistic Input Model

When we study an optimization problem or an algorithm in the model of smoothed analysis, we first need to specify the input model. In the introduction we mentioned as one example the case where the adversary can choose for the knapsack problem for each number in the input an interval of length $1/\phi$ from which it is chosen uniformly at random. This is a special case of one of the input models that we will employ in this lecture. To explain the general model, consider an arbitrary optimization problem whose instances are described completely or in part by some real numbers. This could, for example, be a problem defined on weighted graphs. Then an instance consists of the graph structure and the numbers that describe the weights. It could also be a problem, like the knapsack problem, where there is no combinatorial structure and every instance is completely described by a set of numbers.

In a worst-case analysis the adversary is allowed to choose both the combinatorial structure (if present) and the numbers exactly. In the model we consider, he can still choose the combinatorial structure arbitrarily (e.g., the graph structure) but there is some randomness in the choice of (some of) the numbers. We say that a number is ϕ-*perturbed* if it is a random variable that can be described by a density that is bounded from above by $\phi \geq 1$. A random variable that is chosen from an interval of length $1/\phi$ is one example of a ϕ-perturbed random variable. A Gaussian random variable with standard deviation σ is another example for $\phi = 1/(\sigma\sqrt{2\pi})$. As discussed in the introduction, the parameter ϕ determines the power of the adversary. The larger it is, the more precisely he can specify the input. In the limit for $\phi \to \infty$ the adversary is as powerful as in a worst-case analysis. Usually we assume that ϕ-perturbed numbers take only values between $[-1, 1]$. This normalization is necessary to ensure that the effect of the randomness cannot be ruled out by scaling all numbers in the input.

We will study the (expected) running time of algorithms on inputs with ϕ-perturbed numbers. It is not always necessary to assume that all numbers in the input are ϕ-perturbed. For the knapsack problem, we will consider, for example, instances in which the adversary can determine the profits and the capacity exactly and only the weights are assumed to be ϕ-perturbed numbers. We will call the part of the input that is not perturbed the *deterministic part of the input*. Smoothed analysis can be considered as a worst-case analysis in which the adversary can choose the deterministic part of the input exactly and a density function that is bounded from above by ϕ for every ϕ-perturbed number in the input. The *smoothed running time* of an algorithm is defined to be the worst expected running time the adversary can achieve by the choice of the deterministic part of the input and the density functions, assuming that each ϕ-perturbed number is drawn independently according to the corresponding density.

The smoothed running time is expressed in terms of the input length (where each ϕ-perturbed number is assumed to contribute only one bit to the input length) and the parameter ϕ. In the next chapter, we will present, for example, an algorithm for the knapsack problem whose smoothed running time is $O(n^3\phi)$, where n denotes the number of items and the weights are ϕ-perturbed numbers. It is clear that the smoothed running time has to increase with ϕ because for very large ϕ, the smoothed running time approaches the exponential worst-case running time. Usually, we aim to prove that the smoothed running time of an algorithm is polynomially bounded in the input size and ϕ. This means that already a small amount of randomness suffices to obtain with high probability instances on which the algorithm runs in polynomial time.

Chapter 3

Knapsack Problem and Multiobjective Optimization

The *knapsack problem* is a well-known NP-hard optimization problem. An instance of this problem consists of a set of items, each with a profit and a weight, and a capacity. The goal is to find a subset of the items that maximizes the total profit among all subsets whose total weight does not exceed the capacity. Let $p = (p_1, \dots, p_n)^\mathsf{T} \in \mathbb{R}^n_{\geq 0}$ and $w = (w_1, \dots, w_n)^\mathsf{T} \in \mathbb{R}^n_{\geq 0}$ denote the profits and weights, respectively, and let $W \in \mathbb{R}_{\geq 0}$ denote the capacity. Formally the knapsack problem can be stated as follows:

$$\begin{aligned} \text{maximize} \quad & p^\mathsf{T} x = p_1 x_1 + \cdots + p_n x_n \\ \text{subject to} \quad & w^\mathsf{T} x = w_1 x_1 + \cdots + w_n x_n \leq W, \\ & \text{and } x = (x_1, \dots, x_n)^\mathsf{T} \in \{0,1\}^n. \end{aligned}$$

The knapsack problem has been shown to be NP-hard by Karp in 1972 [16]. Since then it has attracted a great deal of attention, both in theory and in practice. Theoreticians are interested in the knapsack problem because of its simple structure; it can be expressed as a binary program with one linear objective function and one linear constraint. On the other hand, knapsack-like problems often occur in practical applications, and hence practitioners have developed numerous heuristics for solving them. These heuristics work very well on random and real-world instances of the knapsack problem and they usually find optimal solutions quickly even for very large instances.

In Section 3.1, we will present the Nemhauser-Ullmann algorithm for the knapsack problem. This algorithm has an exponential worst-case running time but we will prove in Section 3.2 that its smoothed running time is polynomial. We will discuss generalizations of this analysis to multiobjective optimization problems in Section 3.3. In Section 3.4 we will present the concept of core algorithms for the knapsack problem. These are the fastest known algorithms to solve the knapsack problem in practice and the most successful core algorithm uses the Nemhauser-Ullmann algorithm as a subroutine.

3.1 Nemhauser-Ullmann Algorithm

In the following, we assume that an arbitrary instance $\mathcal{I}$ of the knapsack problem is given. We use the term *solution* to refer to a vector $x \in \{0,1\}^n$, and we say that a solution is *feasible* if $w^\mathsf{T} x \leq W$. We say that a solution x *contains item* i if $x_i = 1$ and that it *does not contain item* i if $x_i = 0$. One naive approach for solving the knapsack problem is to enumerate all feasible solutions and to select the one with maximum profit. This approach is not efficient as there are typically exponentially many feasible solutions. In order to decrease the number of solutions that have to be considered, we observe that a solution x cannot be optimal if there exists another solution y with larger profit and smaller weight than x. We say that such a solution y *dominates* the solution x and observe that it suffices to consider only solutions that are not dominated by other solutions.

Definition 3.1. *A solution* y dominates *a solution* x *if* $p^\mathsf{T} y \geq p^\mathsf{T} x$ *and* $w^\mathsf{T} y \leq w^\mathsf{T} x$ *and if at least one of these inequalities is strict. A solution* x *is called* Pareto-optimal *if it is not dominated by any other solution* y*. The* Pareto set *is the set of all Pareto-optimal solutions.*

Let us remark that the capacity of the knapsack is irrelevant for Definition 3.1 because we do not require the solutions x and y to be feasible. Nemhauser and Ullmann [26] proposed an algorithm for computing the Pareto set of a given knapsack instance. Once the Pareto set is known, the instance can be solved optimally in time linear in the size of this set due to the following observation.

Lemma 3.2. *There always exists an optimal solution that is also Pareto-optimal.*

Proof. Let x be an arbitrary optimal solution for the given instance of the knapsack problem. If x is not Pareto-optimal, then there exists a solution y that dominates x. This means $p^\mathsf{T} y \geq p^\mathsf{T} x$ and $w^\mathsf{T} y \leq w^\mathsf{T} x$ and at least one of these inequalities is strict. The solution y is feasible because $w^\mathsf{T} y \leq w^\mathsf{T} x \leq W$, where the second inequality follows from the feasibility of x. Hence, solution y cannot have a larger profit than solution x because x is an optimal solution. Since $p^\mathsf{T} y \geq p^\mathsf{T} x$, this implies that $p^\mathsf{T} y = p^\mathsf{T} x$. This in turn implies $w^\mathsf{T} y < w^\mathsf{T} x$ because y dominates x. In summary solution y is also an optimal solution with smaller weight than x. This situation is shown in Figure 3.1.

Now either y is a Pareto-optimal solution or by the same reasoning as above there exists another optimal solution z with a smaller weight than y. If we repeat this construction as long as the current solution is not Pareto-optimal, we obtain a sequence of optimal solutions with strictly decreasing weights. As there is only a finite number of solutions, this sequence must be finite as well. The last solution in this sequence is an optimal solution that is also Pareto-optimal. □

We denote the Pareto set by $\mathcal{P} \subseteq \{0,1\}^n$. If this set is known, the solution

$$x^\star = \arg\max_{x \in \mathcal{P}} \{p^\mathsf{T} x \mid w^\mathsf{T} x \leq W\},$$

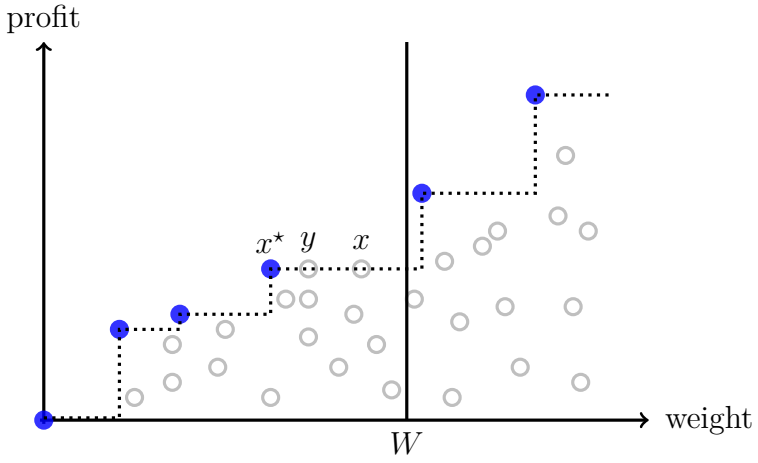


Figure 3.1: In this figure, every solution $z \in \{0,1\}^n$ is depicted as a circle at coordinates $(w^\mathsf{T} z, p^\mathsf{T} z)$. Only the filled circles correspond to Pareto-optimal solutions and only solutions to the left of W are feasible. The solutions x, y, and $x^\star$ are optimal.

which can be found in time linear in the size of $\mathcal{P}$, is an optimal solution of the given instance of the knapsack problem due to the previous lemma. If the arg max is not uniquely determined but a set of multiple solutions, then it follows from the definition of $\mathcal{P}$ that all these solutions must have the same weight and the same profit. In this case $x^\star$ can be chosen as an arbitrary solution from the arg max-set.

The Nemhauser-Ullmann algorithm for computing the Pareto set $\mathcal{P}$ is based on dynamic programming. For each $i \in \{0,1,\ldots,n\}$ it computes the Pareto set $\mathcal{P}_i$ of the modified instance $\mathcal{I}_i$ of the knapsack problem that contains only the first i items of the given instance $\mathcal{I}$. Then $\mathcal{P}_n = \mathcal{P}$ is the set we are looking for. Let

$$\mathcal{S}_i = \{x \in \{0,1\}^n \mid x_{i+1} = \ldots = x_n = 0\}$$

denote the set of solutions that do not contain the items $i+1,\ldots,n$. Formally, solutions of the instance $\mathcal{I}_i$ are binary vectors of length i. We will, however, represent them as binary vectors of length n from $\mathcal{S}_i$. For a solution $x \in \{0,1\}^n$ and an item $i \in \{1,\ldots,n\}$ we denote by x^{+i} the solution that is obtained by adding item i to solution x:

$$x_j^{+i} = \begin{cases} x_j & \text{if } j \neq i, \\ 1 & \text{if } j = i. \end{cases}$$

Furthermore, for a set $\mathcal{S} \subseteq \{0,1\}^n$ of solutions let

$$\mathcal{S}^{+i} = \{y \in \{0,1\}^n \mid \exists x \in \mathcal{S} : y = x^{+i}\}.$$

If for some $i \in \{1,\ldots,n\}$, the set $\mathcal{P}_{i-1}$ is known, the set $\mathcal{P}_i$ can be computed with the help of the following lemma.

Lemma 3.3. *For every $i \in \{1,\ldots,n\}$, the set $\mathcal{P}_i$ is a subset of $\mathcal{P}_{i-1} \cup \mathcal{P}_{i-1}^{+i}$.*

Proof. Let $x \in \mathcal{P}_i$. Based on the value of x_i we distinguish the following two cases.

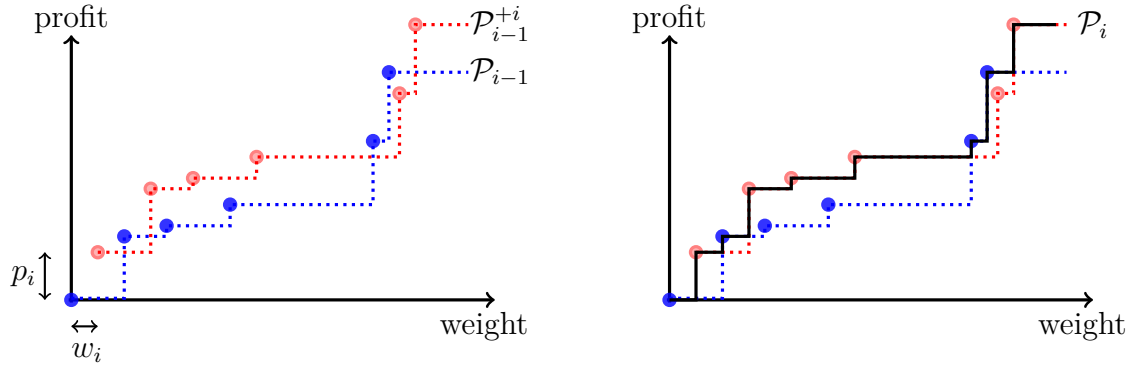


Figure 3.2: Illustration of one iteration of the for loop of the Nemhauser-Ullmann algorithm: The set $\mathcal{P}_{i-1}^{+i}$ is a copy of the set $\mathcal{P}_{i-1}$ that is shifted by (w_i, p_i). The set $\mathcal{P}_i$ is obtained by removing dominated solutions.

First we consider the case $x_i = 0$. We claim that in this case $x \in \mathcal{P}_{i-1}$. Assume for contradiction that $x \notin \mathcal{P}_{i-1}$. Then there exists a solution $y \in \mathcal{P}_{i-1} \subseteq \mathcal{S}_{i-1} \subseteq \mathcal{S}_i$ that dominates x. Since $y \in \mathcal{S}_i$, solution x cannot be Pareto-optimal among the solutions in $\mathcal{S}_i$. Hence, $x \notin \mathcal{P}_i$, contradicting the choice of x.

Now we consider the case $x_i = 1$. We claim that in this case $x \in \mathcal{P}_{i-1}^{+i}$. Since $x \in \mathcal{S}_i$ and $x_i = 1$, there exists a solution $y \in \mathcal{S}_{i-1}$ such that $x = y^{+i}$. We need to show that $y \in \mathcal{P}_{i-1}$. Assume for contradiction that there exists a solution $z \in \mathcal{P}_{i-1}$ that dominates y. Then $p^\mathsf{T} z \geq p^\mathsf{T} y$ and $w^\mathsf{T} z \leq w^\mathsf{T} y$ and one of these inequalities is strict. By adding item i to the solutions y and z, we obtain $p^\mathsf{T} z^{+i} \geq p^\mathsf{T} y^{+i}$ and $w^\mathsf{T} z^{+i} \leq w^\mathsf{T} y^{+i}$ and one of these inequalities is strict. Hence, the solution z^{+i} dominates the solution $x = y^{+i}$. Since $z^{+i} \in \mathcal{S}_i$, this implies $x \notin \mathcal{P}_i$, contradicting the choice of x. □

Due to the previous lemma, the Pareto set $\mathcal{P}_i$ can be computed easily if the Pareto set $\mathcal{P}_{i-1}$ is already known. For this one only needs to compute the set $\mathcal{P}_{i-1} \cup \mathcal{P}_{i-1}^{+i}$ and remove solutions from this set that are dominated by other solutions from this set. Using additionally that $\mathcal{P}_0 = \mathcal{S}_0 = \{0^n\}$, we obtain the following algorithm to solve the knapsack problem.

Nemhauser-Ullmann algorithm

$\mathcal{P}_0 := \{0^n\}$;
for $i = 1, \ldots, n$ **do**
    $\mathcal{Q}_i := \mathcal{P}_{i-1} \cup \mathcal{P}_{i-1}^{+i}$;
    $\mathcal{P}_i := \{x \in \mathcal{Q}_i \mid \nexists y \in \mathcal{Q}_i \colon y \text{ dominates } x\}$;
return $x^\star := \arg\max_{x \in \mathcal{P}_n} \{p^\mathsf{T} x \mid w^\mathsf{T} x \leq W\}$;

The Nemhauser-Ullmann algorithm is illustrated in Figure 3.2.

We analyze the running time of the Nemhauser-Ullmann algorithm using the model of a unit-cost RAM. In this model, arithmetic operations like adding and comparing two numbers can be performed in constant time regardless of their bit-lengths. We use

this model for the sake of simplicity and in order to keep the focus on the important details of the running time analysis.

Theorem 3.4. *The Nemhauser-Ullmann algorithm solves the knapsack problem optimally. There exists an implementation with running time* $\Theta(\sum_{i=0}^{n-1} |\mathcal{P}_i|)$.

Proof. The correctness of the algorithm follows immediately from the previous discussion. In order to achieve the claimed running time, we do not compute the sets $\mathcal{P}_i$ explicitly, but only the values of the solutions in these sets. That is, instead of $\mathcal{P}_i$ only the set $\text{val}(\mathcal{P}_i) := \{(p^\mathsf{T}x, w^\mathsf{T}x) \mid x \in \mathcal{P}_i\}$ is computed. Analogously to the computation of $\mathcal{P}_i$, one can compute $\text{val}(\mathcal{P}_i)$ easily if $\text{val}(\mathcal{P}_{i-1})$ is known. If we store for each element of $\text{val}(\mathcal{P}_i)$ a pointer to the element of $\text{val}(\mathcal{P}_{i-1})$ from which it originates, then in Step 5 the solution $x^\star$ can be efficiently reconstructed from the sets $\text{val}(\mathcal{P}_i)$ and these pointers. We leave the details of this reconstruction as an exercise to the reader.

The running times of Steps 1 and 5 are $O(1)$ and $O(n + |\mathcal{P}|)$, respectively, where the term n accounts for the running time of reconstructing the solution $x^\star$ once its value $(p^\mathsf{T}x^\star, w^\mathsf{T}x^\star)$ is determined. In every iteration i of the for loop, the running time of Step 3 to compute $\text{val}(\mathcal{Q}_i)$ is $O(|\mathcal{P}_{i-1}|)$ because on a unit-cost RAM the set $\text{val}(\mathcal{P}_{i-1}^{+i})$ can be computed in time $O(|\mathcal{P}_{i-1}|)$ from the set $\text{val}(\mathcal{P}_{i-1})$. In a straightforward implementation, the running time of Step 4 is $\Theta(|\mathcal{Q}_i|^2) = \Theta(|\mathcal{P}_{i-1}|^2)$ because we need to compare every pair of values from $\text{val}(\mathcal{Q}_i)$ and each comparison takes time $O(1)$.

Step 4 can be implemented more efficiently. For this, we store the values in each set $\text{val}(\mathcal{P}_i)$ sorted in non-decreasing order of weights. If $\text{val}(\mathcal{P}_{i-1})$ is sorted in this way, then, without any additional computational effort, the computation of the set $\text{val}(\mathcal{Q}_i)$ in Step 3 can be implemented such that $\text{val}(\mathcal{Q}_i)$ is also sorted: The sorted set $\text{val}(\mathcal{P}_{i-1}^{+i})$ can be computed in time $\Theta(|\mathcal{P}_{i-1}|)$. Then, in order to compute the set $\text{val}(\mathcal{Q}_i)$, only the two sorted sets $\text{val}(\mathcal{P}_{i-1})$ and $\text{val}(\mathcal{P}_{i-1}^{+i})$ need to be merged in time $\Theta(|\mathcal{P}_{i-1}|)$. If the set $\text{val}(\mathcal{Q}_i)$ is sorted, Step 4 can be implemented to run in time $\Theta(|\mathcal{Q}_i|)$ as a sweep algorithm going once through $\text{val}(\mathcal{Q}_i)$ in non-decreasing order of weights. □

We advise the reader to think about the details of the previously discussed implementation and to actually implement the Nemhauser-Ullmann algorithm so that it achieves the running time claimed in Theorem 3.4.

3.2 Number of Pareto-optimal Solutions

The running time of the Nemhauser-Ullmann algorithm is polynomially bounded in the size of the instance and the size of the Pareto sets. Hence, the algorithm runs in polynomial time on instances with a polynomial number of Pareto-optimal solutions. It is, however, easy to construct instances of the knapsack problem with exponentially many Pareto-optimal solutions. Thus, not surprisingly, the Nemhauser-Ullmann algorithm does not run in polynomial time in the worst case. On the other hand, it works well in practice and it can solve instances with thousands of items efficiently.

In this section, we study the number of Pareto-optimal solutions in the model of smoothed analysis. In light of the discussion in Section 2.5, it would be reasonable to consider instances of the knapsack problem in which the profits, the weights, and the capacity are ϕ-perturbed numbers. It turns out, however, that we do not even need that much randomness for our analysis. In fact, we assume that only the weights are ϕ-perturbed numbers, while the profits and the capacity are chosen adversarially. This makes the adversary stronger but it suffices to prove a polynomial bound on the smoothed number of Pareto-optimal solutions. Let us remark that all results that we prove in the following would also be true if the weights and the capacity are adversarial, while the profits are ϕ-perturbed numbers.

3.2.1 Upper Bound

In this section, we will prove an upper bound on the smoothed number of Pareto-optimal solutions. We assume in the following that the profits $p_1, \dots, p_n$ are arbitrary and that the weights $w_1, \dots, w_n$ are arbitrary ϕ-perturbed numbers from the interval $[0, 1]$. That is, the adversary can choose for each weight w_i a density function $f_i : [0, 1] \to [0, \phi]$ according to which w_i is chosen independently of the other weights. The restriction of the weights to the interval $[0, 1]$ is only a scaling issue and without loss of generality.

The proof of the following theorem is due to Beier et al. [4]. Their result is much more general, but we present only the following simplified version in this lecture. We discuss some extensions of it in Section 3.3.

Theorem 3.5 ([4])**.** *Let the profits $p_1, \dots, p_n$ be arbitrarily chosen and let the weights $w_1, \dots, w_n$ be arbitrary ϕ-perturbed numbers from the interval $[0, 1]$. Then the expected number of solutions $x \in \{0, 1\}^n$ that are Pareto-optimal with respect to the objective functions $p^\mathsf{T}x$ and $w^\mathsf{T}x$ is bounded from above by $n^2\phi + 1$.*

Proof. We denote the set of Pareto-optimal solutions by $\mathcal{P}$. Every solution $x \in \{0, 1\}^n$ has a weight $w^\mathsf{T}x$ in the interval $[0, n]$ because each weight w_i lies in the interval $[0, 1]$. We partition the interval $(0, n]$ uniformly into $k \in \mathbb{N}$ intervals $I_0^k, \dots, I_{k-1}^k$ for some large number k to be chosen later. Formally, let $I_i^k = (ni/k, n(i + 1)/k]$.

We denote by X^k one plus the number of intervals I_i^k for which there exists at least one Pareto-optimal solution $x \in \mathcal{P}$ with $w^\mathsf{T}x \in I_i^k$. The term $+1$ comes from the solution 0^n, which is always Pareto-optimal and does not belong to any interval I_i^k. Nevertheless, the variable X^k can be much smaller than $|\mathcal{P}|$ because many Pareto-optimal solutions can lie in the same interval I_i^k. The main idea of the proof is that if the subintervals I_i^k are sufficiently small (that is, if k is sufficiently large), then it is very unlikely that there exists a subinterval that contains more than one Pareto-optimal solution. If every subinterval contains at most one Pareto-optimal solution, then $|\mathcal{P}| = X^k$.

In the following, we make this argument more formal. For $k \in \mathbb{N}$, let $\mathcal{F}_k$ denote the event that there exist two different solutions $x, y \in \{0, 1\}^n$ with $|w^\mathsf{T}x - w^\mathsf{T}y| \le n/k$.

Lemma 3.6. *For every $k \in \mathbb{N}$, $\mathbf{Pr}[\mathcal{F}_k] \leq \frac{2^{2n+1}n\phi}{k}$.*

Proof. There are at most 2^{2n} choices for x and y. We prove the lemma by a union bound over all these choices. Let $x, y \in \{0,1\}^n$ with $x \neq y$ be fixed. Then there exists an index i with $x_i \neq y_i$. Assume without loss of generality that $x_i = 0$ and $y_i = 1$. We use the principle of deferred decisions and assume that all weights w_j except for w_i are already revealed. Then $w^\mathsf{T}x - w^\mathsf{T}y = \kappa - w_i$ for some constant κ that depends on x and y and the revealed weights w_j. It holds

$$\mathbf{Pr}\left[|w^\mathsf{T}x - w^\mathsf{T}y| \leq \frac{n}{k}\right] \leq \mathbf{Pr}\left[|\kappa - w_i| \leq \frac{n}{k}\right] = \mathbf{Pr}\left[w_i \in \left[\kappa - \frac{n}{k}, \kappa + \frac{n}{k}\right]\right] \leq \frac{2n\phi}{k},$$

where the last inequality follows from Lemma 2.7 because the density of w_i is bounded from above by ϕ. Now the union bound over all choices for x and y concludes the proof. □

The following lemma is the main building block in the proof of the theorem.

Lemma 3.7. *For every $k \in \mathbb{N}$, $\mathbf{E}\left[X^k\right] \leq n^2\phi + 1$.*

Proof. Let X_i^k denote a random variable that is 1 if the interval I_i^k contains at least one Pareto-optimal solution and 0 otherwise. Then

$$X^k = 1 + \sum_{i=0}^{k-1} X_i^k$$

and by linearity of expectation

$$\mathbf{E}\left[X^k\right] = \mathbf{E}\left[1 + \sum_{i=0}^{k-1} X_i^k\right] = 1 + \sum_{i=0}^{k-1} \mathbf{E}\left[X_i^k\right]. \tag{3.1}$$

Since X_i^k is a random variable that takes only the values 0 or 1, its expected value can be written as follows:

$$\mathbf{E}\left[X_i^k\right] = 0 \cdot \mathbf{Pr}\left[X_i^k = 0\right] + 1 \cdot \mathbf{Pr}\left[X_i^k = 1\right] = \mathbf{Pr}\left[X_i^k = 1\right] = \mathbf{Pr}\left[\exists x \in \mathcal{P} \mid w^\mathsf{T}x \in I_i^k\right]. \tag{3.2}$$

With the help of the following lemma, whose proof is given further below, we can conclude the proof of the lemma.

Lemma 3.8. *For every $t \geq 0$ and every $\varepsilon > 0$,*

$$\mathbf{Pr}\left[\exists x \in \mathcal{P} \mid w^\mathsf{T}x \in (t, t+\varepsilon]\right] \leq n\phi\varepsilon.$$

Lemma 3.8 and (3.2) imply

$$\mathbf{E}\left[X_i^k\right] \leq \frac{n^2\phi}{k}.$$

Together with (3.1) this implies

$$\mathbf{E}\left[X^k\right] = 1 + \sum_{i=0}^{k-1} \mathbf{E}\left[X_i^k\right] \leq 1 + k \cdot \frac{n^2\phi}{k} = n^2\phi + 1.$$ □

With the help of the previous lemmas, we can finish the proof of the theorem as follows:

$$\begin{aligned}
\mathbf{E}\left[|\mathcal{P}|\right] &\overset{(1)}{=} \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i\right] \Big) \\
&\overset{(2)}{=} \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \wedge \mathcal{F}_k\right] + i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \wedge \neg\mathcal{F}_k\right] \Big) \\
&\overset{(3)}{=} \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[\mathcal{F}_k\right] \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[X^k = i \wedge \neg\mathcal{F}_k\right] \Big) \\
&\overset{(4)}{\leq} \mathbf{Pr}\left[\mathcal{F}_k\right] \cdot \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[X^k = i\right] \Big) \\
&\overset{(5)}{\leq} \frac{2^{2n+1} n\phi}{k} \cdot \sum_{i=0}^{2^n} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \mathbf{E}\left[X^k\right] \\
&\overset{(6)}{\leq} \frac{2^{3n+1} n\phi}{k} + n^2\phi + 1. \qquad (3.3)
\end{aligned}$$

Let us comment on the steps in the previous calculation.

- (1) follows from the definition of the expected value.
- (2) follows from the law of total probability.
- The rewriting of the first term in (3) follows from the definition of the conditional probability and the rewriting of the second term follows because $X^k = |\mathcal{P}|$ when the event $\neg\mathcal{F}_k$ occurs.
- (4) follows because $\mathbf{Pr}\left[A \cap B\right] \leq \mathbf{Pr}\left[A\right]$ for all events A and B.
- (5) follows from Lemma 3.6 and the definition of the expected value.
- (6) follows from the identity $\sum_{i=0}^{2^n} \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] = 1$ and Lemma 3.7.

Since (3.3) holds for every $k \in \mathbb{N}$, it must be $\mathbf{E}\left[|\mathcal{P}|\right] \leq n^2\phi + 1$, which proves the theorem. □

It only remains to prove Lemma 3.8.

Proof of Lemma 3.8. First of all we define a random variable $\Lambda(t)$ such that

$$\Lambda(t) \leq \varepsilon \iff \exists x \in \mathcal{P} \colon w^\mathsf{T} x \in (t, t+\varepsilon]. \qquad (3.4)$$

In order to define $\Lambda(t)$, we define the *winner* $x^\star$ to be the most valuable solution satisfying $w^\mathsf{T} x \leq t$, i.e.,

$$x^\star = \arg\max\{p^\mathsf{T} x \mid x \in \{0,1\}^n \text{ and } w^\mathsf{T} x \leq t\}.$$

For $t \geq 0$, such a solution $x^\star$ must always exist. We say that a solution x is a *loser* if it has a higher profit than $x^\star$. By the choice of $x^\star$, losers do not satisfy the constraint $w^\mathsf{T} x \leq t$ (hence their name). We denote by $\hat{x}$ the loser with the smallest weight (see Figure 3.3), i.e.,

$$\hat{x} = \arg\min\{w^\mathsf{T} x \mid x \in \{0,1\}^n \text{ and } p^\mathsf{T} x > p^\mathsf{T} x^\star\}.$$

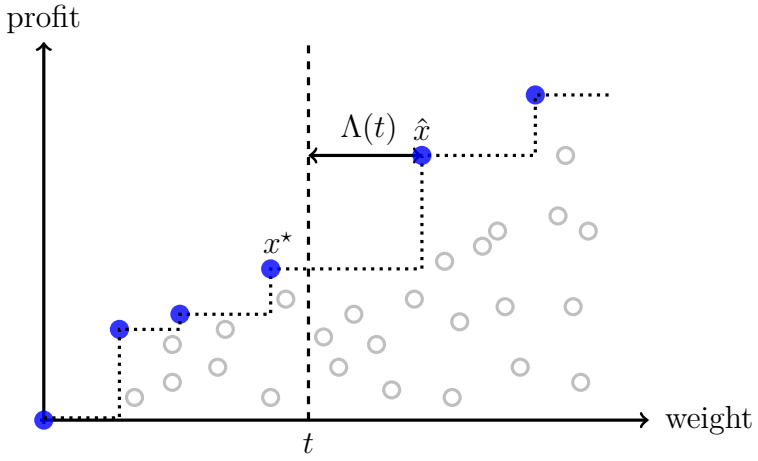


Figure 3.3: Definitions of the winner $x^\star$, the loser $\hat{x}$, and the random variable $\Lambda(t)$.

If there does not exist a solution x with $p^\mathsf{T}x > p^\mathsf{T}x^\star$, then $\hat{x}$ is undefined, which we denote by $\hat{x} = \perp$. Based on $\hat{x}$, we define the random variable $\Lambda(t)$ as

$$\Lambda(t) = \begin{cases} w^\mathsf{T}\hat{x} - t & \text{if } \hat{x} \neq \perp, \\ \perp & \text{if } \hat{x} = \perp. \end{cases}$$

Assume that there exists a Pareto-optimal solution whose weight lies in $(t, t+\varepsilon]$, and let y denote the Pareto-optimal solution with the smallest weight in $(t, t+\varepsilon]$. Then $y = \hat{x}$ and hence $\Lambda(t) = w^\mathsf{T}\hat{x} - t \in (0, \varepsilon]$. Conversely, if $\Lambda(t) \leq \varepsilon$, then $\hat{x}$ must be a Pareto-optimal solution whose weight lies in the interval $(t, t+\varepsilon]$. Together this yields Equivalence (3.4). Hence,

$$\mathbf{Pr}\left[\exists x \in \mathcal{P} \mid w^\mathsf{T}x \in (t, t+\varepsilon]\right] = \mathbf{Pr}\left[\Lambda(t) \leq \varepsilon\right]. \tag{3.5}$$

It only remains to bound the probability that $\Lambda(t)$ does not exceed ε. In order to analyze this probability, we define a set of auxiliary random variables such that $\Lambda(t)$ is guaranteed to always take a value also taken by at least one of the auxiliary random variables. Then we analyze the auxiliary random variables and use a union bound to conclude the desired bound for $\Lambda(t)$. Let $i \in [n]$ be fixed. For $j \in \{0, 1\}$, we define

$$\mathcal{S}^{x_i=j} = \{x \in \{0,1\}^n \mid x_i = j\},$$

and we define $x^{\star,i}$ to be

$$x^{\star,i} = \arg\max\{p^\mathsf{T}x \mid x \in \mathcal{S}^{x_i=0} \text{ and } w^\mathsf{T}x \leq t\}.$$

That is, $x^{\star,i}$ is the winner among the solutions that do not contain item i. We restrict our attention to losers that contain item i and define

$$\hat{x}^i = \arg\min\{w^\mathsf{T}x \mid x \in \mathcal{S}^{x_i=1} \text{ and } p^\mathsf{T}x > p^\mathsf{T}x^{\star,i}\}.$$

If there does not exist a solution $x \in \mathcal{S}^{x_i=1}$ with $p^\mathsf{T}x > p^\mathsf{T}x^{\star,i}$, then $\hat{x}^i$ is undefined, i.e., $\hat{x}^i = \perp$. Based on $\hat{x}^i$, we define the random variable $\Lambda^i(t)$ as

$$\Lambda^i(t) = \begin{cases} w^\mathsf{T}\hat{x}^i - t & \text{if } \hat{x}^i \neq \perp, \\ \perp & \text{if } \hat{x}^i = \perp. \end{cases}$$

Summarizing, $\Lambda^i(t)$ is defined similarly to $\Lambda(t)$, but only solutions that do not contain item i are eligible as winners and only solutions that contain item i are eligible as losers.

Lemma 3.9. *For every choice of profits and weights, either $\Lambda(t) = \bot$ or there exists an index $i \in [n]$ such that $\Lambda(t) = \Lambda^i(t)$.*

Proof. Assume that $\Lambda(t) \neq \bot$. Then there exist a winner $x^\star$ and a loser $\hat{x}$. Since $x^\star \neq \hat{x}$, there must be an index $i \in [n]$ with $x^\star_i \neq \hat{x}_i$. Since all weights are positive and $w^\mathsf{T} x^\star < w^\mathsf{T} \hat{x}$, there must even be an index $i \in [n]$ with $x^\star_i = 0$ and $\hat{x}_i = 1$. We claim that for this index i, $\Lambda(t) = \Lambda^i(t)$. In order to see this, we first observe that $x^\star = x^{\star,i}$. This follows because $x^\star$ is the solution with the highest profit among all solutions with weight at most t. Since it belongs to $\mathcal{S}^{x_i=0}$ it is in particular the solution with the highest profit among all solutions that do not contain item i and have weight at most t. Since $x^\star = x^{\star,i}$, by similar arguments it follows that $\hat{x} = \hat{x}^i$. This directly implies that $\Lambda(t) = \Lambda^i(t)$. □

Lemma 3.10. *For every $i \in [n]$ and every $\varepsilon \geq 0$,*

$$\mathbf{Pr}\left[\Lambda^i(t) \in (0, \varepsilon]\right] \leq \phi\varepsilon.$$

Proof. In order to prove the lemma, it suffices to exploit the randomness of the weight w_i. We apply the principle of deferred decisions and assume that all other weights are fixed arbitrarily. Then the weights of all solutions from $\mathcal{S}^{x_i=0}$ and hence also the solution $x^{\star,i}$ are fixed. If the solution $x^{\star,i}$ is fixed, then also the set of losers $\mathcal{L} = \{x \in \mathcal{S}^{x_i=1} \mid p^\mathsf{T} x > p^\mathsf{T} x^{\star,i}\}$ is fixed. Since, by definition, all solutions from $\mathcal{L}$ contain item i the identity of the solution $\hat{x}^i$ does not depend on w_i. (Of course, the weight $w^\mathsf{T}\hat{x}^i$ depends on w_i. Which solution will become $\hat{x}^i$ is, however, independent of w_i.) This implies that, given the fixed values of the weights w_j with $j \neq i$, we can rewrite the event $\Lambda^i(t) \in (0, \varepsilon]$ as $w^\mathsf{T}\hat{x}^i - t \in (0, \varepsilon]$ for a fixed solution $\hat{x}^i$. For a constant $\kappa \in \mathbb{R}$ depending on the fixed values of the weights w_j with $j \neq i$, we can rewrite this event as $w_i \in (\kappa, \kappa + \varepsilon]$. By Lemma 2.7, the probability of this event is bounded from above by $\phi\varepsilon$. □

Combining Lemmas 3.9 and 3.10 yields

$$\mathbf{Pr}\left[\Lambda(t) \leq \varepsilon\right] \leq \mathbf{Pr}\left[\exists i \in [n] \colon \Lambda^i(t) \in (0, \varepsilon]\right] \leq \sum_{i=1}^{n} \mathbf{Pr}\left[\Lambda^i(t) \in (0, \varepsilon]\right] \leq n\phi\varepsilon.$$

Together with (3.5) this proves the lemma. □

Theorem 3.5 implies the following result on the running time of the Nemhauser-Ullmann algorithm.

Corollary 3.11. *For an instance of the knapsack problem with n items with arbitrary profits and ϕ-perturbed weights from $[0, 1]$, the expected running time of the Nemhauser-Ullmann algorithm is $O(n^3\phi)$.*

Proof. It follows from Theorem 3.4 that the expected running time of the Nemhauser-Ullmann algorithm is bounded from above by

$$O\left(\mathbf{E}\left[\sum_{i=0}^{n-1}|\mathcal{P}_i|\right]\right),$$

where $\mathcal{P}_i$ denotes the Pareto set of the restricted instance that consists only of the first i items. Using linearity of expectation and Theorem 3.5, we obtain that this term is bounded from above by

$$O\left(\sum_{i=0}^{n-1}\mathbf{E}\left[|\mathcal{P}_i|\right]\right) = O\left(\sum_{i=0}^{n-1}(i^2\phi+1)\right) = O(n^3\phi). \qquad \square$$

3.2.2 Lower Bound

In this section we prove that the upper bound for the smoothed number of Pareto-optimal solutions in Theorem 3.5 is asymptotically tight in terms of n. In the lower bound we assume that there are n items and that the adversary has chosen profits $p_i = 2^i$. We only consider the case $\phi = 1$, in which every weight is chosen uniformly at random from $[0, 1]$. The following theorem is due to Beier and Vöcking [6].

Theorem 3.12 ([6])**.** *Consider an instance of the knapsack problem with n items with profits $p_i = 2^i$ for $i \in [n]$. Let the weights $w_1, \ldots, w_n$ be chosen independently and uniformly at random from $[0, 1]$. Then the expected number of solutions $x \in \{0,1\}^n$ that are Pareto-optimal with respect to the objective functions $p^\mathsf{T}x$ and $w^\mathsf{T}x$ is $(n^2 + 3n)/4 + 1 = \Theta(n^2)$.*

Proof. In the following let $\mathcal{P}_j$ denote the Pareto set of the instance that consists only of the first j items. We have shown in Lemma 3.3 that $\mathcal{P}_j$ is a subset of $\mathcal{P}_{j-1} \cup \mathcal{P}_{j-1}^{+j}$. The choice of profits guarantees that every solution from $\mathcal{P}_{j-1}^{+j}$ has a strictly larger profit than any solution from $\mathcal{P}_{j-1}$ because $p_j > \sum_{i=1}^{j-1} p_i$. This implies that solutions from $\mathcal{P}_{j-1}^{+j}$ cannot be dominated by solutions from $\mathcal{P}_{j-1}$. On the other hand, the solution 0^n is always Pareto-optimal and hence $0^n \in \mathcal{P}_{j-1}$, which implies that $\mathcal{P}_{j-1}^{+j}$ contains the solution that consists only of item j. This solution has a higher profit than any solution from $\mathcal{P}_{j-1}$ and hence it dominates all solutions from $\mathcal{P}_{j-1}$ that have weight at least w_j. Since all solutions in $\mathcal{P}_{j-1}^{+j}$ have weight at least w_j, no solution from $\mathcal{P}_{j-1}$ with weight less than w_j is dominated. This is illustrated in Figure 3.4. Hence we have

$$\mathcal{P}_j = \mathcal{P}_{j-1}^{+j} \cup \{x \in \mathcal{P}_{j-1} \mid w^\mathsf{T}x < w_j\}. \tag{3.6}$$

It follows from (3.6) that $|\mathcal{P}_j| \geq |\mathcal{P}_{j-1}^{+j}| = |\mathcal{P}_{j-1}|$ for every j. We define $Y^j = |\mathcal{P}_j| - |\mathcal{P}_{j-1}|$ to be the increase in the number of Pareto-optimal solutions when item j is added to the instance. Then

$$|\mathcal{P}| = 1 + \sum_{j=1}^{n} Y^j, \tag{3.7}$$

where the term $+1$ comes from the solution 0^n, which is already contained in $\mathcal{P}_0$.

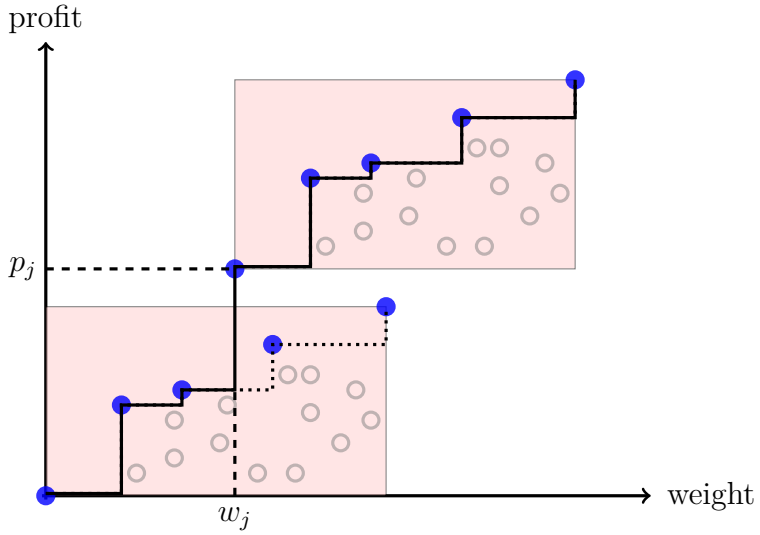


Figure 3.4: Illustration of Equation (3.6): The Pareto set $\mathcal{P}_j$ consists of all solutions from $\mathcal{P}_{j-1}^{+j}$ and all solutions from $\mathcal{P}_{j-1}$ except for the last two because their weights exceed w_j.

In order to analyze the random variables Y^j, we introduce an auxiliary random variable X_α^j for every $j \in [n]$ and every $\alpha \in [0,1]$. It is defined as the number of Pareto-optimal solutions from $\mathcal{P}_j$ with weights in the interval $(0,\alpha)$, i.e.,

$$X_\alpha^j = |\{x \in \mathcal{P}_j \mid w^\mathsf{T} x \in (0,\alpha)\}|.$$

Since $\mathcal{P}_1 = \{0^n, 10^{n-1}\}$, it follows immediately from the previous definition that

$$X_\alpha^1 = \begin{cases} 0 & \text{if } w_1 \geq \alpha, \\ 1 & \text{if } w_1 < \alpha. \end{cases}$$

This implies

$$\mathbf{E}\left[X_\alpha^1\right] = \mathbf{Pr}\left[w_1 < \alpha\right] = \alpha, \tag{3.8}$$

where we used that $\alpha \in [0,1]$ and that w_1 is chosen uniformly at random from $[0,1]$.

For $j \geq 2$, the random variables X_α^j satisfy the following equation, which is illustrated in Figure 3.5:

$$X_\alpha^j = \begin{cases} X_\alpha^{j-1} & \text{if } w_j \geq \alpha, \\ 1 + X_{w_j}^{j-1} + X_{\alpha-w_j}^{j-1} & \text{if } w_j < \alpha. \end{cases} \tag{3.9}$$

In the following, let f_{w_j} denote the density of w_j. Using the law of total expectation, (3.5) implies

$$\begin{aligned} \mathbf{E}\left[X_\alpha^j\right] &= \int_0^1 f_{w_j}(t) \cdot \mathbf{E}\left[X_\alpha^j \mid w_j = t\right] \, dt \\ &= \int_0^1 \mathbf{E}\left[X_\alpha^j \mid w_j = t\right] \, dt \end{aligned}$$

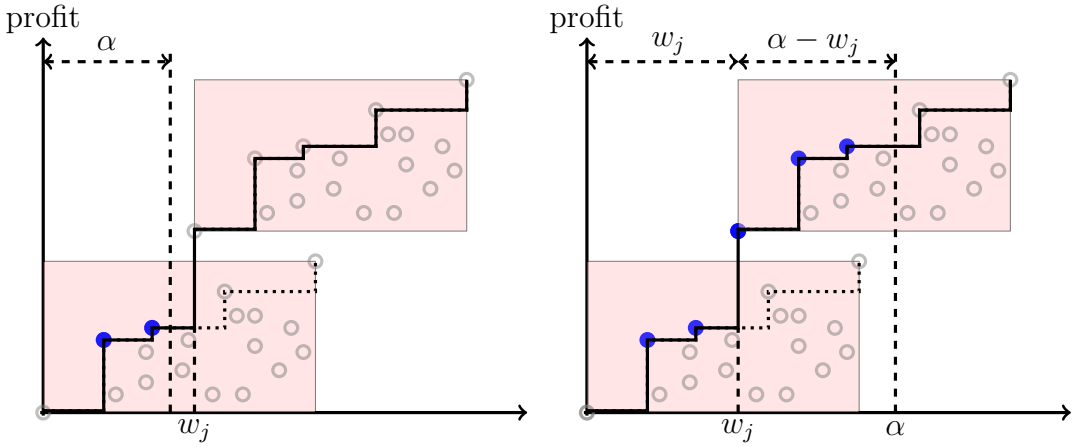


Figure 3.5: Illustration of the two cases in (3.9). In both cases only the blue solutions contribute to X_α^j. Observe that the solution 0^n does not contribute to X_α^{j-1} and X_α^j. The solution that consists only of item j contributes to X_α^j in the case $w_j < \alpha$ and is responsible for the term $+1$.

$$
\begin{aligned}
&= \int_0^\alpha \mathbf{E}\left[X_\alpha^j \mid w_j = t\right] dt + \int_\alpha^1 \mathbf{E}\left[X_\alpha^j \mid w_j = t\right] dt \\
&= \int_0^\alpha \mathbf{E}\left[1 + X_{w_j}^{j-1} + X_{\alpha-w_j}^{j-1} \mid w_j = t\right] dt + \int_\alpha^1 \mathbf{E}\left[X_\alpha^{j-1} \mid w_j = t\right] dt \\
&= \int_0^\alpha \mathbf{E}\left[1 + X_t^{j-1} + X_{\alpha-t}^{j-1}\right] dt + \int_\alpha^1 \mathbf{E}\left[X_\alpha^{j-1}\right] dt \\
&= \int_0^\alpha 1 + \mathbf{E}\left[X_t^{j-1}\right] + \mathbf{E}\left[X_{\alpha-t}^{j-1}\right] dt + (1-\alpha) \cdot \mathbf{E}\left[X_\alpha^{j-1}\right]. \qquad (3.10)
\end{aligned}
$$

In the last line of the previous calculation we integrate over all values t that w_j can take under the condition that $w_j \leq \alpha$. We used that the density of w_j is 1 on $[0, \alpha]$.

Using (3.8) and (3.10) we can prove by induction that $\mathbf{E}[X_\alpha^j] = \alpha j$ for every $\alpha \in [0, 1]$ and every $j \in [n]$. Observe that (3.8) proves the base case $j = 1$. For $j \geq 2$ we obtain with (3.10)

$$
\begin{aligned}
\mathbf{E}\left[X_\alpha^j\right] &= \int_0^\alpha 1 + \mathbf{E}\left[X_t^{j-1}\right] + \mathbf{E}\left[X_{\alpha-t}^{j-1}\right] dt + (1-\alpha) \cdot \mathbf{E}\left[X_\alpha^{j-1}\right] \\
&= \int_0^\alpha 1 + t(j-1) + (\alpha - t)(j-1)\, dt + (1-\alpha) \cdot \alpha(j-1) \\
&= \int_0^\alpha 1 + \alpha(j-1)\, dt + (1-\alpha) \cdot \alpha(j-1) \\
&= \alpha + \alpha \cdot \alpha(j-1) + (1-\alpha) \cdot \alpha(j-1) \\
&= \alpha(j-1) + \alpha = \alpha j.
\end{aligned}
$$

Since $|\mathcal{P}_{j-1}| = |\mathcal{P}_{j-1}^{+j}|$, Equation (3.6) implies

$$Y^j = |\{x \in \mathcal{P}_{j-1} \mid w^\mathsf{T} x < w_j\}| = X_{w_j}^{j-1} + 1.$$

The term $+1$ is due to the solution $0^n \in \mathcal{P}_{j-1}$, which is not counted in $X_{w_j}^{j-1}$. Since the Pareto set $\mathcal{P}_{j-1}$ is independent of the random variable w_j, we can use the law of

total expectation and calculate the expected value of Y^j as follows:

$$\begin{aligned}\mathbf{E}\left[Y^j\right] &= \int_0^1 f_{w_j}(t) \cdot \mathbf{E}\left[Y^j \mid w_j = t\right] \, dt \\ &= \int_0^1 \mathbf{E}\left[Y^j \mid w_j = t\right] \, dt \\ &= \int_0^1 \mathbf{E}\left[X_{w_j}^{j-1} + 1 \mid w_j = t\right] \, dt \\ &= \int_0^1 \mathbf{E}\left[X_t^{j-1} + 1\right] \, dt \\ &= \int_0^1 (j-1)t + 1 \, dt = 1 + (j-1)\int_0^1 t \, dt = 1 + \frac{j-1}{2}.\end{aligned}$$

Using (3.7) we obtain

$$\begin{aligned}\mathbf{E}\left[|\mathcal{P}|\right] = \mathbf{E}\left[1 + \sum_{j=1}^{n} Y^j\right] &= 1 + \sum_{j=1}^{n} \mathbf{E}\left[Y^j\right] = 1 + \sum_{j=1}^{n} \left(1 + \frac{j-1}{2}\right) \\ &= 1 + n + \frac{n(n-1)}{4} = \frac{n^2}{4} + \frac{3n}{4} + 1.\end{aligned}$$

This concludes the proof. □

3.3 Multiobjective Optimization

The upper bound on the expected number of Pareto-optimal solutions that Beier et al. [4] proved is much more general than the version that we have presented in Theorem 3.5.

Theorem 3.13 ([4]). *Let $\mathcal{S} \subseteq \{0,1\}^n$ and $p : \mathcal{S} \to \mathbb{R}$ be arbitrary. Let $w_1, \ldots, w_n$ be arbitrary ϕ-perturbed numbers from the interval $[-1, 1]$. Then the expected number of solutions $x \in \mathcal{S}$ that are Pareto-optimal with respect to the objective functions $p(x)$ and $w^\mathsf{T}x$ is $O(n^2\phi)$.*

The first generalization compared to Theorem 3.5 is that an arbitrary set $\mathcal{S} \subseteq \{0,1\}^n$ of solutions is given. In the case of the knapsack problem, every vector from $\{0,1\}^n$ is a solution, i.e., $\mathcal{S} = \{0,1\}^n$. The second generalization is that the adversarial objective function p does not have to be linear. In fact, it can be an arbitrary function that maps every solution to some real value. The third generalization is that the coefficients w_i are not restricted to positive values anymore. The restriction of the profits to the interval $[-1, 1]$ is only a scaling issue and (almost) without loss of generality. The result holds regardless of whether the objective functions should be maximized or minimized.

We will not prove Theorem 3.13 in this lecture, but let us remark that its proof is very similar to the proof of Theorem 3.5. In fact we never used in that proof that $\mathcal{S} = \{0,1\}^n$ and that p is linear. The fact that all weights w_i are positive was only used to argue

that there must be an index i with $x_i^\star = 0$ and $\hat{x}_i = 1$. For general w_i, it could also be the other way round. Handling this issue is the only modification of the proof that is not completely straightforward.

To illustrate the power of Theorem 3.13, let us discuss its implications on graph problems. For a given graph with m edges $e_1, \dots, e_m$, one can, for example, identify every vector $x \in \{0,1\}^m$ with a subset of edges $E(x) = \{e_i \mid x_i = 1\}$. Then x is the so-called incidence vector of the edge set $E(x)$. If, for example, there is a source node s and a target node t given, one could choose the set $\mathcal{S}$ of feasible solutions as the set of all incidence vectors of paths from s to t in the given graph. This way, Theorem 3.13 implies that the smoothed number of Pareto-optimal paths in the bicriteria shortest-path problem is $O(m^2\phi)$. In this problem, every edge of the graph is assigned a weight and a cost, and one wants to find Pareto-optimal paths from s to t with respect to total weight and total cost. Similarly, one could choose $\mathcal{S}$ as the set of incidence vectors of all spanning trees of a given graph. Then the result implies that there are only $O(m^2\phi)$ Pareto-optimal spanning trees in expectation in the bicriteria spanning tree problem.

These results can even be generalized to optimization problems with one arbitrary objective function and d linear objective functions with ϕ-perturbed coefficients [10]. For these problems the bound becomes $O(n^{2d}\phi^{d(d+1)})$. It can even be improved to $O(n^{2d}\phi^d)$ if all densities are quasiconcave, where a function $f : \mathbb{R} \to \mathbb{R}$ is called quasiconcave if there exists some $x \in \mathbb{R}$ such that f is monotonically increasing on $(-\infty, x]$ and monotonically decreasing on $[x, \infty)$.

3.4 Core Algorithms

Corollary 3.11 is an explanation why the Nemhauser-Ullmann algorithm performs better in practice than predicted by its worst-case running time. However, a running time in the order of n^3 is still not good enough for instances with millions of items. In this section, we will present *core algorithms* for the knapsack problem that have a much better (almost linear) running time. This section is based on an article by Beier and Vöcking [5]

For a given instance of the knapsack problem with n items with profits $p_1, \dots, p_n$, weights $w_1, \dots, w_n$, and capacity W, core algorithms first compute an optimal solution of the *fractional knapsack problem*, in which items are divisible and can be packed fractionally into the knapsack:

$$\begin{aligned} \text{maximize} \quad & p^\mathsf{T} x = p_1 x_1 + \cdots + p_n x_n \\ \text{subject to} \quad & w^\mathsf{T} x = w_1 x_1 + \cdots + w_n x_n \leq W, \\ & \text{and } x = (x_1, \dots, x_n)^\mathsf{T} \in [0,1]^n. \end{aligned}$$

The following greedy algorithm computes an optimal solution of the fractional knapsack problem in time $O(n \log n)$.

Greedy algorithm for fractional knapsack problem

1: Sort the items in descending order by their profit-to-weight ratio p_i/w_i.
   Assume that after this step $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$.
2: $t := W$;
3: **for** $i = 1, \ldots, n$ **do**
4: **if** $w_i \leq t$ **then**
5: $x_i := 1$;
6: $t := t - w_i$;
7: **else**
8: $x_i := t/w_i$;
9: $t := 0$;

It is left as an exercise for the reader to prove that the previous algorithm computes an optimal solution of the fractional knapsack problem. The running time of the algorithm is $O(n \log n)$ due to the sorting in Step 1. It is another exercise for the reader to find an algorithm that solves the fractional knapsack problem in time $O(n)$.

From the pseudocode of the greedy algorithm it is clear that it computes a solution in which all variables x_i except for at most one take values from $\{0, 1\}$: If the remaining capacity t is at least w_i, the variable x_i is set to 1. Otherwise, if $t > 0$ but $t < w_i$ the variable x_i is set to some fractional value from $(0, 1)$. After that $t = 0$ and all remaining variables are set to 0.

It can also happen that the algorithm computes a solution in which all variables have a value from $\{0, 1\}$. Then this solution is also an optimal solution of the non-fractional knapsack problem. Hence, this case is not interesting and we consider in the following only instances in which the greedy algorithm computes a solution $\bar{x} \in [0, 1]^n$ in which exactly one variable takes a fractional value from $(0, 1)$. We call the item corresponding to this variable *break item* and we denote it by $i^\star$. We assume in the following that the items are sorted in descending order by their profit-to-weight ratio p_i/w_i. Then $\bar{x}_i = 1$ for all $i < i^\star$ and $\bar{x}_i = 0$ for all $i > i^\star$.

We call the ray that starts at the origin $(0, 0)$ and goes through $(w_{i^\star}, p_{i^\star})$ *Dantzig ray*. For the sake of simplicity we assume in the following that $i^\star$ is the only item that lies on the Dantzig ray (this assumption is not necessary but it simplifies the presentation a bit). In the following, let r denote the slope of the Dantzig ray. The optimal fractional solution $\bar{x}$ has a simple geometric structure. It contains all items above the Dantzig ray while it does not contain any item below this ray. This is illustrated in Figure 3.6.

Now let $x^\star \in \{0, 1\}^n$ denote an optimal solution of the non-fractional knapsack problem. Core algorithms are based on the observation that typically the solutions $x^\star$ and $\bar{x}$ differ only in a few items. Typically these items are close to the Dantzig ray while $x^\star$ and $\bar{x}$ agree on all items that have a significant distance from the Dantzig ray, i.e., all items significantly above this ray belong to $x^\star$ and all items significantly below this ray do not belong to $x^\star$. We call the set of all items that do not have a significant distance from the Dantzig ray the *core*.

To make this more formal, we first define the *loss* ℓ_i of item i to be its vertical distance from the Dantzig ray, i.e., $\ell_i := |p_i - rw_i|$ (see Figure 3.6). For a solution $x \in \{0, 1\}^n$

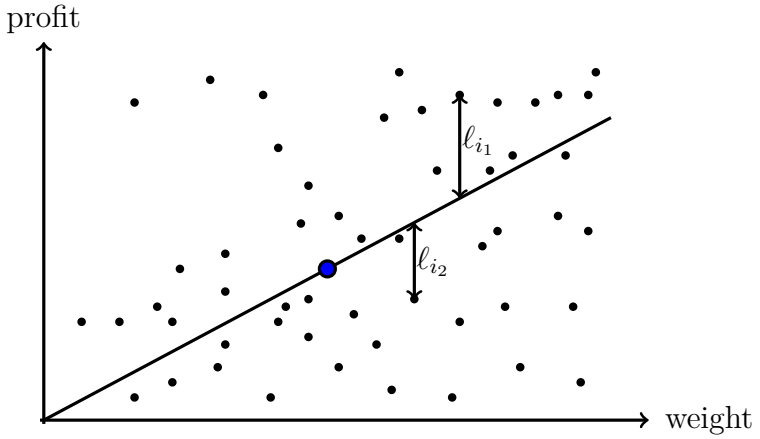


Figure 3.6: In this figure each item i corresponds to a dot at position (w_i, p_i). The break item lies on the Dantzig ray and it is shown larger than the other items. The loss of two items i_1 and i_2 is shown exemplary.

we define $\Delta(x) \subseteq [n]$ to be the set of items on which x and $\bar{x}$ disagree, i.e., $\Delta(x) = \{i \in [n] \mid x_i \neq \bar{x}_i\}$. With the help of these definitions we can express the gap between the profits of the optimal fractional solution $\bar{x}$ and some binary solution $x \in \{0,1\}^n$ as follows.

Lemma 3.14. *Let $x \in \{0,1\}^n$ be an arbitrary solution. Then*

$$p^\mathsf{T}\bar{x} - p^\mathsf{T}x = r(W - w^\mathsf{T}x) + \sum_{i \in \Delta(x)} \ell_i.$$

Intuitively the profit of the solution x can be smaller than the profit of $\bar{x}$ for two reasons. First it might be the case that the solution x does not use the complete capacity $W = w^\mathsf{T}\bar{x}$ of the knapsack. Second all items above the Dantzig ray that x does not contain and all items below the Dantzig ray that x contains also contribute to the gap between the profits of $\bar{x}$ and x. This is captured by the two terms in the sum.

Proof. Let $\Delta_0(x) = \{i \in [n] \mid i < i^\star \text{ and } x_i = 0\}$ and $\Delta_1(x) = \{i \in [n] \mid i > i^\star \text{ and } x_i = 1\}$. Then $\Delta(x) = \Delta_0(x) \cup \Delta_1(x) \cup \{i^\star\}$. For every item $i \in \Delta_0(x)$, we have

$$p_i\bar{x}_i - p_ix_i = p_i = rw_i + \ell_i$$

and for every item $i \in \Delta_1(x)$, we have

$$p_i\bar{x}_i - p_ix_i = -p_i = -(rw_i - \ell_i) = -rw_i + \ell_i.$$

Furthermore,

$$p_{i^\star}\bar{x}_{i^\star} - p_{i^\star}x_{i^\star} = rw_{i^\star}(\bar{x}_{i^\star} - x_{i^\star}).$$

Now taking the sum yields

$$p^\mathsf{T}\bar{x} - p^\mathsf{T}x = \sum_{i=1}^{n}(p_i\bar{x}_i - p_ix_i)$$

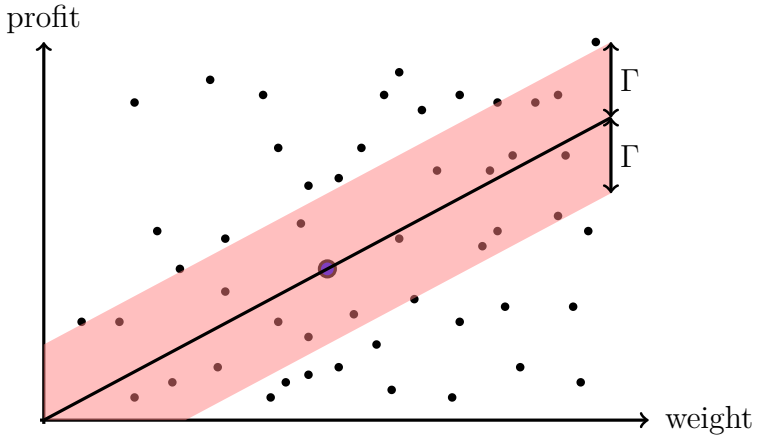


Figure 3.7: The core consists of all items whose loss is at most Γ.

$$
\begin{aligned}
&= \left(\sum_{i\in\Delta_0(x)} rw_i + \ell_i\right) + \left(\sum_{i\in\Delta_1(x)} -rw_i + \ell_i\right) + rw_{i^\star}(\bar{x}_{i^\star} - x_{i^\star}) \\
&= r\left(\left(\sum_{i\in\Delta_0(x)} w_i\right) - \left(\sum_{i\in\Delta_1(x)} w_i\right) + w_{i^\star}(\bar{x}_{i^\star} - x_{i^\star})\right) + \sum_{i\in\Delta_0(x)\cup\Delta_1(x)} \ell_i \\
&= r(w^\mathsf{T}\bar{x} - w^\mathsf{T}x) + \sum_{i\in\Delta(x)} \ell_i \\
&= r(W - w^\mathsf{T}x) + \sum_{i\in\Delta(x)} \ell_i.
\end{aligned}
$$

In the second to last line we used that $\Delta(x) = \Delta_0(x)\cup\Delta_1(x)\cup\{i^\star\}$ and that $\ell_{i^\star} = 0$. □

Assume that we know the *additive integrality gap* Γ of the given instance of the knapsack problem, i.e., $\Gamma = p^\mathsf{T}\bar{x} - p^\mathsf{T}x^\star$, where $\bar{x}$ and $x^\star$ denote the optimal fractional and non-fractional solutions, respectively. Then we could use the previous lemma to define the core C.

Corollary 3.15. *It holds*

$$\Delta(x^\star) \subseteq C := \{i \in [n] \mid \ell_i \leq \Gamma\}.$$

That is, all items with a loss greater than Γ *above the Dantzig ray are contained in* $x^\star$ *while all items with a loss greater than* Γ *below the Dantzig ray are not contained in* $x^\star$.

Proof. Assume that there exists an item i with loss $\ell_i > \Gamma$ and $i \in \Delta(x^\star)$. Then with Lemma 3.14 we obtain the following contradiction

$$\Gamma = p^\mathsf{T}\bar{x} - p^\mathsf{T}x^\star \geq \ell_i > \Gamma.$$ □

The previous corollary is illustrated in Figure 3.7. If Γ is known, then one only needs to solve the following knapsack problem on the items in the core:

$$\text{maximize} \quad \sum_{i\in C} p_i x_i$$

$$\text{subject to} \quad \sum_{i \in C} w_i x_i \leq W', \\ \text{and } \forall i \in C : x_i \in \{0,1\},$$

where W' is defined as the remaining capacity if one takes into account that all items above the Dantzig ray that do not belong to the core are definitely contained in the optimal non-fractional solution $x^\star$, i.e.,

$$W' = W - \sum_{i:\ell_i > \Gamma, i < i^\star} w_i.$$

This way, we have reduced the given instance of the knapsack problem to an instance with potentially fewer items. The hope is that typically there are only a few items in the core such that the remaining instance consists only of a few items. This is indeed observed in practice and proved by Beier and Vöcking [5] for instances in which every weight and every profit is chosen uniformly at random from $[0,1]$.

If the remaining instance consists only of a logarithmic number of items then it can even be solved by a brute-force search in polynomial time. However, the remaining instance can also be solved by the Nemhauser-Ullmann algorithm, which yields an algorithm that runs in almost linear time on random inputs. The following theorem has been proved by Beier and Vöcking [5]. As its proof is rather involved, we will not discuss it in the lecture. We will also not discuss their algorithm in detail and leave it at the main idea as discussed above.

Theorem 3.16 ([5])**.** *There exists a core algorithm, which solves the remaining instance by the Nemhauser-Ullmann algorithm, with expected running time $O(n \log^c n)$ for some constant c on instances with n items in which every profit and every weight is chosen independently and uniformly at random from $[0,1]$ and the capacity is chosen as $W = \beta n$ for some $\beta \in (0, 1/2)$.*

Let us now briefly discuss the issue that our definition of the core requires Γ to be known in advance. This is of course not a practical assumption because Γ is unknown and there is also no easy way to compute it. Most implementations of core algorithms try to estimate Γ. One variant, the *expanding core algorithm*, first chooses the smallest value γ such that the core

$$C(\gamma) := \{i \in [n] \mid \ell_i \leq \gamma\}$$

with respect to γ contains at least one item apart from the break item. Then it computes the optimal solution x with respect to the core $C(\gamma)$, i.e., x is the solution that contains all items above $C(\gamma)$, no item below $C(\gamma)$, and the optimal subset of items from $C(\gamma)$. For $\gamma \geq \Gamma$, the solution x coincides with $x^\star$ according to Corollary 3.15. For smaller γ, the solution x is in general not optimal.

If $p^\mathsf{T}\bar{x} - p^\mathsf{T}x \leq \gamma$, we can be sure that $\gamma \geq \Gamma$, which implies that x is an optimal solution, because

$$\Gamma = p^\mathsf{T}\bar{x} - p^\mathsf{T}x^\star \leq p^\mathsf{T}\bar{x} - p^\mathsf{T}x \leq \gamma.$$

If $p^\mathsf{T}\bar{x} - p^\mathsf{T}x > \gamma$, we increase γ and compute the optimal solution with respect to the enlarged core $C(\gamma)$. This algorithm is shown in the following pseudocode.

Expanding Core Algorithm

1: Compute the optimal fractional solution $\bar{x}$.
2: $\gamma := 0$
3: **repeat**
4: Increase γ by the smallest amount that is necessary for $|C(\gamma)|$ to increase.
5: Compute the optimal solution x with respect to the core $C(\gamma)$.
6: **until** $p^{\mathsf{T}}\bar{x} - p^{\mathsf{T}}x \leq \gamma$
7: **return** x;

Beier and Vöcking [7] have implemented this algorithm. They used the Nemhauser-Ullmann algorithm to compute the optimal solution x with respect to the current core in Line 5. With their implementation random instances with millions of items can be solved optimally within a few seconds.

Chapter 4

Smoothed Complexity of Binary Optimization Problems

We have seen in the previous chapter that the knapsack problem can be solved efficiently on inputs with ϕ-perturbed weights or ϕ-perturbed profits. Instead of studying each problem separately, we will now give a general characterization which combinatorial optimization problems can be solved efficiently on instances with ϕ-perturbed numbers.

We will study *linear binary optimization problems*. In an instance of such a problem Π, a linear objective function $c^\mathsf{T}x = c_1x_1 + \cdots + c_nx_n$ is to be minimized or maximized over an arbitrary set $\mathcal{S} \subseteq \{0,1\}^n$ of feasible solutions. The problem Π could, for example, be the traveling salesman problem and the coefficients c_i could be the edge lengths. (See also the discussion at the end of Section 3.3 on how graph problems can be encoded as binary optimization problems.) One could also encode the knapsack problem as a linear binary optimization problem. Then $\mathcal{S}$ contains all subsets of items whose total weight does not exceed the capacity.

We will study the *smoothed complexity* of linear binary optimization problems, by which we mean the complexity of instances in which the coefficients $c_1, \ldots, c_n$ are ϕ-perturbed numbers from the interval $[-1, 1]$. We will assume without loss of generality that the objective function $c^\mathsf{T}x$ is to be minimized. Since ϕ-perturbed coefficients are real numbers and do not have a finite encoding with probability 1, Turing machines or random-access machines do not seem to be appropriate machine models to study the smoothed complexity of linear binary optimization problems. One could change the input model and assume that the ϕ-perturbed coefficients are discretized by rounding them after a polynomial number, say n^2, of bits. The effect of this rounding is so small that it does not influence our results. We will, however, not make this assumption explicit and use, for the sake of simplicity, the continuous random variables in our probabilistic analysis. In particular, when defining the input size we will not take the encoding length of the coefficients c_i into account. Instead we assume that the coefficients $c_1, \ldots, c_n$ contribute in total only n to the input length.

To state the main results of this chapter, let us recall two definitions from computa-

tional complexity. We call a linear binary optimization problem *strongly* NP*-hard* if it is already NP-hard when restricted to instances with integer coefficients c_i in which the largest absolute value $C := \max_i |c_i|$ of any of the coefficients is bounded by a polynomial in the input length. The TSP is, for example, strongly NP-hard because it is already NP-hard when all edges have length either 1 or 2. The knapsack problem, on the other hand, is not strongly NP-hard because instances in which all profits are integers and polynomially bounded in the input size can be solved by dynamic programming in polynomial time.

A language L belongs to the complexity class ZPP (zero-error probabilistic polynomial time) if there exists a randomized algorithm A that decides for each input x in expected polynomial time whether or not x belongs to L. That is, A always produces the correct answer but its running time is a random variable whose expected value is bounded polynomially for every input x. Let us point out that the expectation is only with respect to the random decisions of the algorithm and not with respect to a randomly chosen input (as usually the case in this lecture). It is yet unclear whether or not P = ZPP. In any case, languages that belong to ZPP are generally considered to be easy to decide.

Theorem 4.1. *Let* Π *be a linear binary optimization problem that is strongly* NP*-hard. Then there does not exist an algorithm for* Π *whose expected running time is polynomially bounded in* N *and* ϕ *for instances with* ϕ*-perturbed coefficients from* $[-1, 1]$*, where* N *denotes the input length, unless* NP $\subseteq$ ZPP*.*

Proof. Let A denote an algorithm that solves ϕ-perturbed instances of Π in expected time $p(N, \phi)$ for some polynomial p, where N denotes the input length. Since Π is strongly NP-hard, there exists a polynomial q such that the problem Π is already NP-hard when restricted to instances with integer coefficients whose absolute values are bounded from above by $q(N)$. We claim that we can solve these instances with the help of A in expected polynomial time. Hence we have found an NP-hard optimization problem that belongs to ZPP. This implies NP $\subseteq$ ZPP (analogously to the fact that the existence of an NP-hard optimization problem that belongs to P implies NP $\subseteq$ P).

Let $\mathcal{I}$ be an instance of Π with input length N and integer coefficients $c_1, \ldots, c_n$ whose absolute values are bounded from above by $q(N)$. First we scale the coefficients such that they lie in the interval $[-1, 1]$. This does not change the optimal solution. In the following we denote by $\tilde{c}_1, \ldots, \tilde{c}_n$ the scaled coefficients, i.e., $\tilde{c}_i = c_i/C$, where $C := \max_i |c_i| \leq q(N)$. Let $\phi := 2nC$. We choose for each coefficient $\tilde{c}_i$ an interval $I_i \subseteq [-1, 1]$ of length $1/\phi$ with $\tilde{c}_i \in I_i$ and choose c'_i uniformly at random from I_i. In this way, we obtain an instance of Π with ϕ-perturbed coefficients. Let us call this instance $\mathcal{I}'$. We can use algorithm A to solve this instance in expected time $p(N, \phi)$. Due to our choice of ϕ and due to $N \geq n$, this expected running time is polynomially bounded in the input length N:

$$p(N, \phi) = p(N, 2nC) \leq p(N, 2nq(N)) \leq p(N, 2Nq(N)) = \text{poly}(N).$$

We will argue that any optimal solution of the ϕ-perturbed instance $\mathcal{I}'$ is also an optimal solution of the instance $\mathcal{I}$. Hence, algorithm A solves not only the ϕ-perturbed

instance $\mathcal{I}'$ but also the instance $\mathcal{I}$ in expected polynomial time. Let x denote the optimal solution of instance $\mathcal{I}'$ that algorithm A computes. We will argue that x is also an optimal solution of the instance $\mathcal{I}$. Assume for contradiction that there exists a better solution $x^\star$ for instance $\mathcal{I}$. Since all coefficients in $\mathcal{I}$ are integers, it must be $c^\mathsf{T}x - c^\mathsf{T}x^\star \geq 1$ and hence

$$\tilde{c}^\mathsf{T}x - \tilde{c}^\mathsf{T}x^\star \geq \frac{1}{C}.$$

Since $|c'_i - \tilde{c}_i| \leq 1/\phi$ for all i, this implies

$$\begin{aligned}(c')^\mathsf{T}x - (c')^\mathsf{T}x^\star &\geq \tilde{c}^\mathsf{T}x - \tilde{c}^\mathsf{T}x^\star - \frac{n}{\phi}\\ &= \tilde{c}^\mathsf{T}x - \tilde{c}^\mathsf{T}x^\star - \frac{1}{2C} \geq \frac{1}{2C} > 0.\end{aligned}$$

Hence, contradicting our choice of x as optimal solution of $\mathcal{I}'$, the solution $x^\star$ is better than x also with respect to the coefficients $c'_1, \ldots, c'_n$. □

The previous theorem shows that ϕ-perturbed instances of strongly NP-hard optimization problems are not easier to solve than worst-case instances. Hence, these problems stay hard also in the model of smoothed analysis. One consequence of this result is that there is no hope that the TSP can be solved efficiently when the edge lengths are randomly perturbed. This is a clear contrast to the knapsack problem, which is easy to solve on randomly perturbed inputs, as shown in the previous chapter. We will now prove a more general positive result. We say that a linear binary optimization problem Π can be solved in *pseudo-linear time* if there exists an algorithm whose running time on instances with integer coefficients is bounded from above by $p(N) \cdot C$, where p denotes a polynomial, N denotes the input length, and C denotes the largest absolute value of any of the coefficients.

Theorem 4.2. *A linear binary optimization problem Π that can be solved in pseudo-linear time in the worst case can be solved in expected polynomial time (with respect to the input length and ϕ) on instances with ϕ-perturbed numbers.*

Proof. There exists a constant k and an algorithm A_{pseudo} that solves integer instances of Π in time $O(N^k \cdot C)$, where N denotes the input length and C denotes the largest absolute value of any of the coefficients. Instances in which all coefficients are rational numbers from $[-1, 1]$ that can each be encoded with at most b bits after the binary point can be solved by the algorithm A_{pseudo} in time $O(N^k \cdot 2^b)$ because if we scale each coefficient in such an instance by 2^b then all coefficients are integers with absolute value at most 2^b.

We will use algorithm A_{pseudo} as a subroutine to obtain an algorithm that solves ϕ-perturbed instances of Π efficiently. The idea of this algorithm is rather simple: round all coefficients of the ϕ-perturbed instance $\mathcal{I}$ after a certain number b of bits after the binary point and solve the rounded instance $\mathcal{I}_b$ in time $O(N^k \cdot 2^b)$ with algorithm A_{pseudo}. If $b = \Theta(\log n)$ then A_{pseudo} runs in polynomial time and the optimal solutions of the instances $\mathcal{I}$ and $\mathcal{I}_b$ coincide with high probability. In order to make this more precise, we will first of all introduce and analyze the *winner gap* Δ.

Given an instance $\mathcal{I}$ of Π with a set $\mathcal{S} \subseteq \{0,1\}^n$ of feasible solutions, the winner gap is defined as

$$\Delta = c^\mathsf{T} x^{\star\star} - c^\mathsf{T} x^\star,$$

where

$$x^\star = \arg\min\{c^\mathsf{T} x \mid x \in \mathcal{S}\} \quad \text{and} \quad x^{\star\star} = \arg\min\{c^\mathsf{T} x \mid x \in \mathcal{S} \setminus \{x^\star\}\}$$

denote the best and second best solution of $\mathcal{I}$, respectively. If $\mathcal{I}$ contains multiple optimal solutions, then $\Delta = 0$. We will prove that in ϕ-perturbed instances the winner gap is typically not too small.

Lemma 4.3 (Isolation Lemma)**.** *Let $\mathcal{I}$ be an instance of Π with ϕ-perturbed coefficients $c_1, \dots, c_n$. Then for every $\varepsilon > 0$,*

$$\mathbf{Pr}\left[\Delta \leq \varepsilon\right] \leq 2n\phi\varepsilon.$$

Proof. The proof of this lemma is similar to the proof of Lemma 3.8. Hence, we present it in a rather condensed version and refer the reader to the proof of Lemma 3.8 for more verbose explanations.

We first define a set of auxiliary random variables $\Delta_1, \dots, \Delta_n$ with the property that at least one of these auxiliary random variables coincides with Δ. We define

$$x^{i=0} = \arg\min\{c^\mathsf{T} x \mid x \in \mathcal{S} \text{ and } x_i = 0\},$$
$$x^{i=1} = \arg\min\{c^\mathsf{T} x \mid x \in \mathcal{S} \text{ and } x_i = 1\},$$

and $\Delta_i = |c^\mathsf{T} x^{i=0} - c^\mathsf{T} x^{i=1}|$.

Let $i \in [n]$ be an index in which the best and the second best solution differ, i.e., $x_i^\star \neq x_i^{\star\star}$. For this index i it holds either $x^{i=0} = x^\star$ and $x^{i=1} = x^{\star\star}$ or $x^{i=1} = x^\star$ and $x^{i=0} = x^{\star\star}$. In any case $\Delta = \Delta_i$.

For any i, we would like to give an upper bound on the probability that $\Delta_i \leq \varepsilon$. We use again the principle of deferred decisions and assume that all c_j with $j \neq i$ are already fixed arbitrarily. Then also the solutions $x^{i=0}$ and $x^{i=1}$ are already fixed as their identity does not depend on the yet unknown coefficient c_i. This implies that $\Delta_i = |\kappa - c_i|$ for some constant κ that depends on the coefficients c_j with $j \neq i$. Hence,

$$\mathbf{Pr}\left[\Delta_i \leq \varepsilon\right] \leq \mathbf{Pr}\left[|\kappa - c_i| \leq \varepsilon\right] \leq 2\phi\varepsilon,$$

where we used that c_i is a random variable whose density is bounded from above by ϕ.

Altogether we obtain

$$\mathbf{Pr}\left[\Delta \leq \varepsilon\right] \leq \mathbf{Pr}\left[\exists i \in [n] : \Delta_i \leq \varepsilon\right] \leq \sum_{i=1}^{n} \mathbf{Pr}\left[\Delta_i \leq \varepsilon\right] \leq 2n\phi\varepsilon. \qquad \square$$

For a coefficient c_i we denote by $\lfloor c_i \rfloor_b$ and $\lceil c_i \rceil_b$ the numbers that are obtained when c_i is rounded down or up after b bits after the binary point, respectively. That is, $\lfloor c_i \rfloor_b \leq c_i$

and $\lceil c_i \rceil_b \geq c_i$. It holds $|\lfloor c_i \rfloor_b - c_i| \leq 2^{-b}$ and $|\lceil c_i \rceil_b - c_i| \leq 2^{-b}$. Let $\lfloor c \rfloor_b$ and $\lceil c \rceil_b$ denote the vectors that consist of the rounded coefficients.

For a vector $a \in \mathbb{R}^n$, let $\mathrm{opt}(a)$ denote the optimal solution of the given instance of Π with objective function $a^\mathsf{T} x$, i.e., $\mathrm{opt}(a) = \arg\min\{a^\mathsf{T} x \mid x \in \mathcal{S}\}$. The isolation lemma implies the following corollary.

Corollary 4.4. *For any $b \in \mathbb{N}$, and any vector c' with $c'_i \in \{\lfloor c_i \rfloor_b, \lceil c_i \rceil_b\}$ for any $i \in [n]$,*

$$\mathbf{Pr}\left[\mathrm{opt}(c') \neq \mathrm{opt}(c)\right] \leq 2^{-b+2} n^2 \phi.$$

Proof. For every $i \in [n]$, $|c_i - c'_i| \leq 2^{-b}$. Hence, for any $x \in \{0,1\}^n$

$$|c^\mathsf{T} x - (c')^\mathsf{T} x| \leq n 2^{-b}.$$

Hence, if $\Delta > 2n2^{-b}$, then the optimal solution $x^\star = \mathrm{opt}(c)$ is also optimal with respect to c'. This implies

$$\mathbf{Pr}\left[\mathrm{opt}(c') \neq \mathrm{opt}(c)\right] \leq \mathbf{Pr}\left[\Delta \leq 2n2^{-b}\right] \leq 2^{-b+2} n^2 \phi,$$

where the last inequality follows from Lemma 4.3. □

Corollary 4.4 immediately yields an algorithm that always runs in polynomial time and solves ϕ-perturbed instances of Π with high probability correctly: use the pseudo-linear algorithm A_{pseudo} to compute $\mathrm{opt}(\lfloor c \rfloor_b)$ in time $O(N^k \cdot 2^b)$. For $b = \log(n^3\phi) + 2$ the running time of A_{pseudo} is polynomial and $\mathrm{opt}(c) = \mathrm{opt}(\lfloor c \rfloor_b)$ with probability at least $1 - 1/n$. However, in the following we aim for an algorithm that always computes the optimal solution and whose expected running time is polynomial.

For this, we add another step to the algorithm: After we have computed $\mathrm{opt}(\lfloor c \rfloor_b)$, we test whether or not it is also an optimal solution with respect to the objective function $c^\mathsf{T} x$. If this is not the case, then we increase the precision b by one and compute $\mathrm{opt}(\lfloor c \rfloor_b)$ for the increased precision. This is repeated until the solution $\mathrm{opt}(\lfloor c \rfloor_b)$ is also optimal with respect to the objective function $c^\mathsf{T} x$. Now the question is of course, how one can efficiently test if the solution $\mathrm{opt}(\lfloor c \rfloor_b)$ is also optimal with respect to the objective function $c^\mathsf{T} x$. For this, we use the following method Certify(c,x,b).

Certify(c,x,b)

1: For $i \in [n]$, let $\tilde{c}_i = \begin{cases} \lfloor c_i \rfloor_b & \text{if } x_i = 0, \\ \lceil c_i \rceil_b & \text{if } x_i = 1. \end{cases}$
2: Compute $y = \mathrm{opt}(\tilde{c})$ with algorithm A_{pseudo}.
3: **if** $x = y$ **then**
4: **return true**;
5: **else**
6: **return false**;

Lemma 4.5. *If the method Certify(c,x,b) returns true, then x is an optimal solution with respect to the objective function $c^\mathsf{T} x$.*

Proof. For contradiction, assume that Certify(c,x,b) returns true and that there exists a solution $z \in \mathcal{S}$ with $c^\mathsf{T} z < c^\mathsf{T} x$. Let $i \in [n]$ be an index with $z_i \neq x_i$. If $z_i = 1$ and $x_i = 0$, then

$$\tilde{c}_i x_i - \tilde{c}_i z_i = c_i x_i - \tilde{c}_i z_i = c_i x_i - \lfloor c_i \rfloor_b z_i \geq c_i x_i - c_i z_i.$$

If $z_i = 0$ and $x_i = 1$, then

$$\tilde{c}_i x_i - \tilde{c}_i z_i = \tilde{c}_i x_i - c_i z_i = \lceil c_i \rceil_b x_i - c_i z_i \geq c_i x_i - c_i z_i.$$

Hence, the definition of $\tilde{c}$ guarantees that for every $i \in [n]$, $\tilde{c}_i x_i - \tilde{c}_i z_i \geq c_i x_i - c_i z_i$. This implies

$$\tilde{c}^\mathsf{T} x - \tilde{c}^\mathsf{T} z \geq c^\mathsf{T} x - c^\mathsf{T} z > 0.$$

Hence, x is not an optimal solution with respect to $\tilde{c}$ and Certify(c,x,b) does not return true. □

Lemma 4.6. *Certify(c,x,b) returns false with probability at most* $2^{-b+2} n^2 \phi$.

Proof. The argument is similar to the argument that we used in the proof of Corollary 4.4. If $\Delta > 2^{-b+1} n$, then $\text{opt}(c') = \text{opt}(c)$ for any vector c' with $c'_i = \lfloor c_i \rfloor_b$ or $c'_i = \lceil c_i \rceil_b$ for any $i \in [n]$, which implies $\text{opt}(\lfloor c \rfloor_b) = \text{opt}(c) = \text{opt}(\tilde{c})$. According to Lemma 4.3, $\mathbf{Pr}\left[\Delta \leq 2^{-b+1} n\right] \leq 2^{-b+2} n^2 \phi$. □

Based on the method Certify(c,x,b), we can devise the following algorithm AdaptiveRounding that solves every instance $\mathcal{I}$ of Π optimally.

AdaptiveRounding($\mathcal{I}$)

1: **for** $b = 1, \ldots, n$ **do**
2: Compute $x = \text{opt}(\lfloor c \rfloor_b)$ with algorithm A_{pseudo}.
3: **if** Certify(c,x,b) **then return** x;
4: Compute $x = \text{opt}(c)$ by brute force.
5: **return** x;

The correctness of this algorithm follows immediately from Lemma 4.5. It remains to analyze its expected running time. Let X denote the random variable that corresponds to the final value of b in the algorithm, i.e., X denotes the number of iterations of the for-loop. Lemma 4.6 implies that

$$\mathbf{Pr}\left[X \geq j\right] \leq 2^{-j+3} n^2 \phi$$

because X can only be at least j if Certify returns false in iteration $j-1$. Furthermore, let t_j denote the total running time of iteration j of this loop. Since in this iteration two calls to algorithm A_{pseudo} are made with coefficients that have j bits each, $t_j = O(N^k \cdot 2^j)$. We assume in the following that the time to solve $\mathcal{I}$ by brute force is $O(2^n)$. Depending on the problem, it might actually be $O(2^n)$ times a polynomial in N, but we ignore this additional polynomial for the sake of simplicity because it has

no significant effect on the analysis. Let T denote the random variable that describes the total running time of the algorithm. Then

$$\begin{aligned}\mathbf{E}[T] &= \sum_{j=1}^{n} t_j \cdot \mathbf{Pr}[X \geq j] + O(2^n) \cdot \mathbf{Pr}[X \geq n] \\ &\leq \sum_{j=1}^{n} O(N^k \cdot 2^j) \cdot 2^{-j+3} n^2 \phi + O(2^n) \cdot 2^{-n+3} n^2 \phi \\ &= O(N^k n^3 \phi) = O(N^{k+3} \phi).\end{aligned}$$

In summary, the algorithm AdaptiveRounding always computes an optimal solution of the given instance $\mathcal{I}$ and its expected running time on ϕ-perturbed instances is polynomially bounded in N and ϕ. □

The results that we presented in this section can be found in [8]. Theorems 4.1 and 4.2 do not give a complete characterization of the smoothed complexity of linear binary optimization problems. They leave open the smoothed complexity of problems that can be solved in pseudo-polynomial but not pseudo-linear time in the worst case. One can show that also these problems can be solved in expected polynomial time on ϕ-perturbed instances [29], which completes the characterization.

Chapter 5

Successive Shortest Path Algorithm

We study the *Successive Shortest Path (SSP) algorithm* for the *minimum-cost flow problem*. This chapter is based on an article by Brunsch et al. [9].

Let us first define the minimum-cost flow problem. A *flow network* is a simple directed graph $G = (V, E)$ together with a *capacity function* $u\colon E \to \mathbb{R}_{\geq 0}$, a source $s \in V$, and a sink $t \in V$. For convenience, we assume that there are no directed cycles of length two. In the minimum-cost flow problem there is an additional *cost function* $c\colon E \to [0, 1]$ given. A *flow* for such an instance is a function $f\colon E \to \mathbb{R}_{\geq 0}$ that obeys the capacity constraints $0 \leq f(e) \leq u(e)$ for all edges $e \in E$ and Kirchhoff's law, i.e., $\sum_{e=(u,v)\in E} f(e) = \sum_{e'=(v,w)\in E} f(e')$ for all nodes $v \in V \setminus \{s, t\}$. A *maximum flow* f is a flow with maximum *value*

$$|f| = \sum_{e=(s,v)\in E} f(e) - \sum_{e=(v,s)\in E} f(e).$$

The cost of a flow is defined as $c(f) = \sum_{e\in E} f(e) \cdot c(e)$. In the *minimum-cost flow problem* the goal is to find a cheapest maximum flow, a so-called *minimum-cost flow*.

The SSP algorithm is a simple greedy algorithm to solve the minimum-cost flow problem, which is known to have an exponential worst-case running time [33]. An experimental study of Kovács [19] shows, however, that the SSP algorithm performs well in practice and is much faster than most polynomial-time algorithms for the minimum-cost flow problem. We will explain why the SSP algorithm comes off so well by applying the framework of smoothed analysis.

5.1 The SSP Algorithm

Let $G = (V, E)$ be a flow network. For a pair $e = (u, v)$, we denote by e^{-1} the pair (v, u). Observe that $e \in E$ implies $e^{-1} \notin E$ because we assume that there are no cycles of length two. For a flow f, the *residual network* G_f is the directed graph with vertex set V, arc set $E' = E_\mathrm{f} \cup E_\mathrm{b}$, where

$$E_\mathrm{f} = \left\{e \,|\, e \in E \text{ and } f(e) < u(e)\right\}$$

is the set of so-called *forward arcs* and

$$E_{\mathrm{b}} = \left\{ e^{-1} \,|\, e \in E \text{ and } f(e) > 0 \right\}$$

is the set of so-called *backward arcs*, a capacity function $u' \colon E' \to \mathbb{R}$, defined by

$$u'(e) = \begin{cases} u(e) - f(e) & \text{if } e \in E\,, \\ f(e^{-1}) & \text{if } e^{-1} \in E\,, \end{cases}$$

and a cost function $c' \colon E' \to \mathbb{R}$, defined by

$$c'(e) = \begin{cases} c(e) & \text{if } e \in E\,, \\ -c(e^{-1}) & \text{if } e^{-1} \in E\,. \end{cases}$$

Let f be a flow, let G_f denote the corresponding residual network, and let P denote a path from s to t in G_f. We say that the flow f' is obtained from the flow f *by augmenting* $\delta \geq 0$ *units of flow along path* P if for all $e \in E$

$$f'(e) = \begin{cases} f(e) + \delta & \text{if } e \in P \cap E_f\,, \\ f(e) - \delta & \text{if } e^{-1} \in P \cap E_b\,, \\ f(e) & \text{otherwise}\,. \end{cases}$$

In the SSP algorithm, which is given in the following pseudocode, we choose δ as large as possible subject to the constraint that f' is a flow (i.e., $0 \leq f'(e) \leq u(e)$ for all $e \in E$).

Successive Shortest Path Algorithm

1: start with the empty flow $f_0 = 0$
2: **while** G_{f_i} contains an s-t path **do**
3: find a shortest s-t path P_i in G_{f_i} with respect to the arc costs
4: augment the flow as much as possible along path P_i to obtain a new flow f_{i+1}

5.1.1 Elementary Properties

In this section, we summarize a few elementary properties of the SSP algorithm that we will need in our analysis.

Theorem 5.1 ([18]). *In any round i, flow f_i is a flow with minimum cost among all flows with value $|f_i|$. In particular, the last flow obtained by the SSP algorithm is a minimum-cost flow.*

As a consequence, no residual network G_{f_i} contains a directed cycle with negative total costs. Otherwise, we could augment along such a cycle to obtain a flow with the same value as f_i but smaller costs. In particular, this implies that the shortest paths in G_{f_i} from s to nodes $v \in V$ form a shortest path tree rooted at s.

Next, we observe that for any node v the distances in the residual network from s to v and from v to t are monotonically increasing.

Lemma 5.2. *Let $d_i(v)$ denote the distance from s to node v in the residual network G_{f_i} and let $d'_i(v)$ denote the distance from node v to t in the residual network G_{f_i}. Then the sequences $d_0(v)$, $d_1(v)$, $d_2(v)$, ... and $d'_0(v)$, $d'_1(v)$, $d'_2(v)$, ... are monotonically increasing for every $v \in V$.*

Proof. We only prove the statement for the sequence $d_0(v), d_1(v), d_2(v), \ldots$. The statement for the sequence $d'_0(v), d'_1(v), d'_2(v), \ldots$ can be shown analogously. Let $i \geq 0$ be an arbitrary integer. We show $d_i(v) \leq d_{i+1}(v)$ by induction on the depth of node v in the shortest path tree T_{i+1} of the residual network $G_{f_{i+1}}$ rooted at s. For the root s, the claim holds since $d_i(s) = d_{i+1}(s) = 0$. Now assume that the claim holds for all nodes up to a certain depth k, consider a node v with depth $k+1$, and let u denote its parent. Consequently, $d_{i+1}(v) = d_{i+1}(u) + c(e)$ for $e = (u, v)$. If arc e has been available in G_{f_i}, then $d_i(v) \leq d_i(u) + c_e$. If not, then the SSP algorithm must have augmented along e^{-1} in step $i + 1$ to obtain flow f_{i+1} and, hence, $d_i(u) = d_i(v) + c(e^{-1}) = d_i(v) - c(e)$ (because e^{-1} is part of the shortest-path-tree in G_{f_i}). In both cases the inequality $d_i(v) \leq d_i(u) + c(e)$ holds. By the induction hypothesis for node u, we obtain $d_i(v) \leq d_i(u) + c(e) \leq d_{i+1}(u) + c(e) = d_{i+1}(v)$. □

Lemma 5.2 implies in particular that the distance from the source s to the sink t is monotonically increasing, which yields the following corollary.

Corollary 5.3. *Let $i < j$, let P_i denote the shortest path in G_{f_i}, and let P_j denote the shortest path in G_{f_j}. Then $c(P_i) \leq c(P_j)$.*

5.2 Smoothed Analysis

In the following, we consider instances of the minimum-cost flow problem in which the costs are ϕ-perturbed numbers from $[0, 1]$. That is, the adversary specifies for each edge e a probability density function $f_e \colon [0, 1] \to [0, \phi]$ according to which the cost $c(e)$ is randomly drawn independently of the other edge costs. Furthermore, the adversary chooses the network graph $G = (V, E)$ and the edge capacities. The edge capacities are even allowed to be real values. Let $n = |V|$ and $m = |E|$. We define the smoothed running time of the SSP algorithm as the worst expected running time the adversary can achieve and we prove the following theorem.

Theorem 5.4. *The SSP algorithm requires $O(mn\phi)$ augmentation steps in expectation and its smoothed running time is $O(mn\phi(m + n \log n))$.*

This theorem can be seen as an explanation for why it is unlikely to encounter instances on which the SSP algorithm requires exponentially many steps. The following theorem, which we will not prove in this lecture, shows that the upper bound given in the previous theorem is asymptotically tight for $\phi = \Omega(n)$.

Theorem 5.5 ([9])**.** *For given positive integers n, $m \in \{n, \ldots, n^2\}$, and $\phi \leq 2^n$ there exists a minimum-cost flow network with $O(n)$ nodes, $O(m)$ edges, and ϕ-perturbed edge costs on which the SSP algorithm requires $\Omega(m \cdot \min\{n, \phi\} \cdot \phi)$ augmentation steps with probability 1.*

5.2.1 Terminology and Notation

Consider the execution of the SSP algorithm on the flow network G. We denote the set $\{f_0, f_1, \ldots, f_N\}$ of all flows encountered by the SSP algorithm by $\mathsf{F}(G)$. Furthermore, we denote by $\mathcal{P}(G) = \{P_0, P_1, \ldots, P_{N-1}\}$ the paths that the SSP algorithm augments along, i.e., P_i is a shortest path in the residual network G_{f_i}. We omit the parameter G if it is clear from the context and we say that f_i *induces* P_i. In the following, we will determine the expected size of $\mathcal{P}(G)$ because this equals the expected number of augmentation steps of the SSP algorithm.

To distinguish between the original network G and some residual network G_f, we refer to the edges in the residual network as *arcs*, whereas we refer to the edges in the original network as *edges*. For a given arc e in a residual network G_f, we denote by e_0 the corresponding edge in the original network G, i.e., $e_0 = e$ if $e \in E$ (i.e., e is a forward arc) and $e_0 = e^{-1}$ if $e \notin E$ (i.e., e is a backward arc). An arc e is called *empty* in a flow f if e belongs to G_f, but e^{-1} does not. Empty arcs e are either forward arcs that do not carry flow or backward arcs whose corresponding edge e_0 carries as much flow as possible. We say that an arc *becomes saturated* (during an augmentation) when it is contained in the current augmenting path, but it does not belong to the residual network that we obtain after this augmentation.

In the remainder, a *path* is always a simple directed path. Let P be a path, and let u and v be contained in P in this order. By $u \overset{P}{\rightsquigarrow} v$, we refer to the sub-path of P starting from node u going to node v, by $\overleftarrow{P}$ we refer to the path we obtain by reversing the direction of each arc of P. We denote by

$$c(P) = \sum_{e \in P \cap E_f} c(e) - \sum_{e \in P \cap E_b} c(e^{-1})$$

the cost of P. We will also refer to $c(P)$ as the *length* of the path P. We call any flow network G' a *possible residual network* of G if there is a flow f for G such that $G' = G_f$. Paths in possible residual networks are called *possible paths*.

5.2.2 Proof of the Upper Bound

The idea and structure of the proof of Theorem 5.4 are very similar to the proof of Theorem 3.5. Since all edges have costs from $[0, 1]$, the cost $c(P)$ of any path $P \in \mathcal{P}$ lies in the interval $[0, n]$. Similarly in the proof of Theorem 3.5, every Pareto-optimal solution from $\mathcal{P}$ is mapped to some number from $[0, n]$ by the weight function $w^\mathsf{T}x$. The idea to count the number of Pareto-optimal solutions was to divide the interval $[0, n]$ into small subintervals of length n/k for some k. We argued that if k is large enough, then with high probability there are no two Pareto-optimal solutions that are mapped to the same subinterval. Hence, in order to count the number of Pareto-optimal solutions, it suffices to count the number of non-empty subintervals. The approach to prove Theorem 5.4 is the same, except that we do not count Pareto-optimal solutions but paths encountered by the SSP algorithm. The main difference is the analysis of the

probability that a given subinterval contains a path from $\mathcal{P}$, which is more challenging than its counterpart in the proof of Theorem 3.5.

For $k \in \mathbb{N}$, let $\mathcal{F}_k$ denote the event that there exist two nodes $u \in V$ and $v \in V$ and two distinct possible u-v paths whose lengths differ by at most n/k.

Lemma 5.6. *For every $k \in \mathbb{N}$, $\mathbf{Pr}[\mathcal{F}_k] \leq \frac{2n^{2n+1}\phi}{k}$.*

Proof. Fix two nodes u and v and two distinct possible u-v paths P_1 and P_2. Then there is an edge e such that one of the paths – without loss of generality path P_1 – contains arc e or e^{-1}, but the other one contains neither of them. If we use the principle of deferred decisions and fix all edge costs except for the cost of edge e, then the length of P_2 is already determined whereas the length of P_1 depends linearly on the cost $c(e)$. Hence, $c(e)$ must fall into a fixed interval of length $2n/k$ in order for the path lengths of P_1 and P_2 to differ by at most n/k. The probability for this is bounded by $2n\phi/k$ because $c(e)$ is chosen according to a density function that is bounded from above by ϕ. There are at most n^n different simple possible paths. A union bound over all pairs of such paths concludes the proof. □

The following lemma, which we prove further below, is the crucial ingredient in the proof of Theorem 5.4.

Lemma 5.7. *For every $d \geq 0$ and every $\varepsilon > 0$,*

$$\mathbf{Pr}[\exists P \in \mathcal{P} \mid c(P) \in (d, d+\varepsilon]] \leq 2m\varepsilon\phi.$$

Now the proof of Theorem 5.4 follows along the same lines as the proof of Theorem 3.5. We partition the interval $(0, n]$ uniformly into $k \in \mathbb{N}$ intervals $I_0^k, \ldots, I_{k-1}^k$ for some large number k to be chosen later. Formally, let $I_i^k = (ni/k, n(i+1)/k]$.

We denote by X^k the number of intervals I_i^k for which there exists at least one path $P \in \mathcal{P}$ with $c(P) \in I_i^k$. If the event $\mathcal{F}_k$ does not occur, then $|\mathcal{P}| = X^k$. (Observe that with probability 1 there does not exist a possible s-t path of length 0).

Lemma 5.8. *For every $k \in \mathbb{N}$, $\mathbf{E}\left[X^k\right] \leq 2mn\phi$.*

Proof. Let X_i^k denote a random variable that is 1 if there exists a path $P \in \mathcal{P}$ with $c(P) \in I_i^k$ and 0 otherwise. Then

$$X^k = \sum_{i=0}^{k-1} X_i^k$$

and by linearity of expectation

$$\mathbf{E}\left[X^k\right] = \mathbf{E}\left[\sum_{i=0}^{k-1} X_i^k\right] = \sum_{i=0}^{k-1} \mathbf{E}\left[X_i^k\right] = \sum_{i=0}^{k-1} \mathbf{Pr}\left[\exists P \in \mathcal{P} \mid c(P) \in I_i^k\right]. \tag{5.1}$$

Lemma 5.7 and (5.1) imply

$$\mathbf{E}\left[X^k\right] = \sum_{i=0}^{k-1} \mathbf{Pr}\left[\exists P \in \mathcal{P} \mid c(P) \in I_i^k\right] \leq \sum_{i=0}^{k-1} \frac{2mn\phi}{k} = 2mn\phi. \qquad \square$$

The last missing ingredient in the proof of Theorem 5.4 is a bound on the number of augmentation steps of the SSP algorithm in the worst case. While in Chapter 3 it was clear that there can be at most 2^n Pareto-optimal solutions, the situation is slightly more complicated here. Lemma 5.6 shows in particular (in the limit for $k \to \infty$) that any two different possible paths have different lengths with probability 1. Hence, we will assume throughout this chapter that the following property holds.

Property 5.9. *For all nodes u and v the lengths of all possible u-v paths are pairwise distinct.*

Assuming this property, we can bound the number of augmentation steps of the SSP algorithm in the worst case.

Lemma 5.10. *The number $|\mathcal{P}|$ of augmentation steps of the SSP algorithm is bounded from above by 3^m.*

Proof. Consider two paths $P_i \in \mathcal{P}$ and $P_{i+1} \in \mathcal{P}$ encountered by the SSP algorithm. Since at least one arc of P_i is not contained in the residual network $G_{f_{i+1}}$, the paths P_i and P_{i+1} must be different. Now Corollary 5.3 and Property 5.9 imply $c(P_i) < c(P_{i+1})$. Hence the lengths of the augmenting paths that the SSP algorithm encounters are strictly increasing. In particular, no path is encountered more than once by the SSP algorithm.

We say that two residual networks are equivalent if they contain the same arcs (possibly with different capacities). Since the shortest path in two equivalent residual networks is the same, at most one residual network from each equivalence class is encountered by the SSP algorithm. The number of equivalence classes is bounded by 3^m because for every edge $e \in E$ there are three possibilities: either e and e^{-1} are both contained in the residual network or exactly one of them. This completes the proof. □

Proof of Theorem 5.4. The theorem is implied by the following calculation:

$$\begin{aligned}
\mathbf{E}\left[|\mathcal{P}|\right] &= \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i\right] \Big) \\
&= \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \wedge \mathcal{F}_k\right] + i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \wedge \neg\mathcal{F}_k\right] \Big) \\
&= \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[\mathcal{F}_k\right] \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[X^k = i \wedge \neg\mathcal{F}_k\right] \Big) \\
&\leq \mathbf{Pr}\left[\mathcal{F}_k\right] \cdot \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[X^k = i\right] \Big) \\
&\leq \frac{2n^{2n+1}\phi}{k} \cdot \sum_{i=0}^{3^m} \Big(i \cdot \mathbf{Pr}\left[|\mathcal{P}| = i \mid \mathcal{F}_k\right] \Big) + \mathbf{E}\left[X^k\right] \\
&\leq \frac{2n^{2n+1}3^m\phi}{k} + 2mn\phi. \qquad (5.2)
\end{aligned}$$

For a justification of the steps in this calculation, see the proof of Theorem 3.5.

Since (5.2) holds for every $k \in \mathbb{N}$, it must be $\mathbf{E}[|\mathcal{P}|] \leq 2mn\phi$, which proves the first part of the theorem. Since each step of the SSP algorithm runs in time $O(m + n \log n)$ using Dijkstra's algorithm (see, e.g., Korte [18] for details), also the statement about the running time of the SSP algorithm follows. □

5.2.3 Further Properties of the SSP Algorithm

We will now introduce further properties of the SSP algorithm that we will need in the proof of Lemma 5.7.

By $\mathscr{C}$ we denote the function $\mathscr{C} : [0, |f_N|] \to \mathbb{R}_{\geq 0}$ that maps any $x \in [0, |f_N|]$ to the cost of the cheapest flow f with value x, i.e., $\mathscr{C}(x) = \min\{c(f) \mid |f| = x\}$, where the minimum is taken over all flows f with value x.

Lemma 5.11. *The function $\mathscr{C}$ is continuous, monotonically increasing, and piecewise linear. The break points of $\mathscr{C}$ are the values of the flows $f_i \in \mathsf{F} \setminus \{f_0, f_N\}$ (assuming Property 5.9).*

Proof. The proof follows from Theorem 5.1 and the observation that the cost of the flow is linearly increasing when gradually increasing the flow along the shortest path in the residual network until at least one arc becomes saturated. The slope of the cost function is given by the length of that path. □

Example 5.12. *Consider the flow network depicted in Figure 5.1. The cost $c(e)$ and the capacity $u(e)$ of an edge e are given by the notation $c(e), u(e)$. For each step of the SSP algorithm, Figure 5.3 lists the relevant part of the augmenting path (excluding s, s', t', and t), its length, the amount of flow that is sent along that path, and the arcs that become saturated. As can be seen in the table, the values $|f|$ of the encountered flows $f \in \mathsf{F}$ are 0, 2, 3, 5, 7, 10, and 12. Except for 0 and 12, these are the breakpoints of the cost function $\mathscr{C}$, and the lengths of the augmenting paths equal the slopes of $\mathscr{C}$ (see Figure 5.2).*

Another important property that we will use in the proof of Lemma 5.7 is the following.

Lemma 5.13. *In any step of the SSP algorithm, any s-t path in the residual network contains at least one empty arc.*

Proof. The claim is true for the empty flow f_0. Now consider a flow $f_{i+1} \in \mathsf{F} \setminus \{f_0\}$, its predecessor flow f_i, the path P_i, which is a shortest path in the residual network G_{f_i}, and an arbitrary s-t path P in the current residual network $G_{f_{i+1}}$. The paths P_i and P are different because at least one arc of P_i is not contained in $G_{f_{i+1}}$ anymore. We show that at least one arc in P is empty.

For this, fix one arc $e = (x, y)$ from P_i that is not contained in the current residual network $G_{f_{i+1}}$ since it became saturated by the augmentation along P_i. Let v be the first node of P that occurs in the sub-path $y \overset{P_i}{\rightsquigarrow} t$ of P_i, and let u be the last node in the sub-path $s \overset{P}{\rightsquigarrow} v$ of P that belongs to the sub-path $s \overset{P_i}{\rightsquigarrow} x$ of P_i (see Figure 5.4).

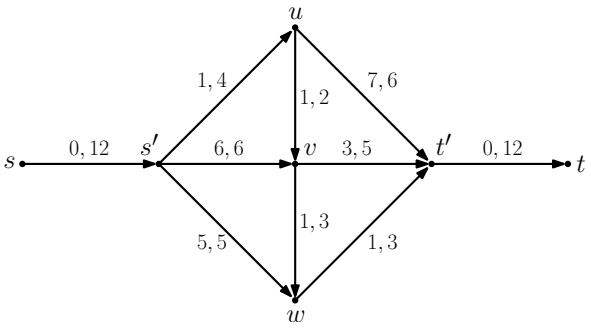


Figure 5.1: Minimum-cost flow network with source s and sink t.

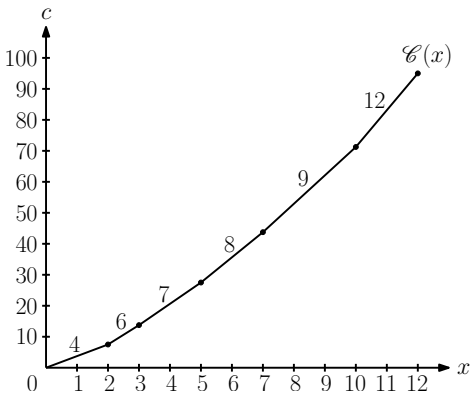


Figure 5.2: Cost function $\mathscr{C}$.

step	1	2	3	4	5	6
path	u, v, w	w	w, v	u	v	v, u
path length	4	6	7	8	9	12
amount of flow	2	1	2	2	3	2
saturated arcs	(u, v)	(w, t')	(w, v)	(s', u)	(v, t')	(v, u)

Figure 5.3: The augmenting paths for Example 5.12.

By the choice of u and v, all nodes on the sub-path $P' = u \overset{P}{\rightsquigarrow} v$ of P except for u and v do not belong to P_i. Hence, the arcs of P' are also available in the residual network G_{f_i} and have the same capacity in both residual networks G_{f_i} and $G_{f_{i+1}}$.

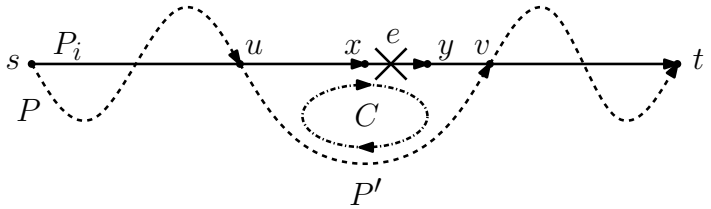


Figure 5.4: Paths P and P_i in the residual network G_{f_i}.

In the remainder of this proof, we show that at least one arc of P' is empty. Assume to the contrary that none of the arcs is empty in $G_{f_{i+1}}$ and, hence, in G_{f_i}. This implies that, for each arc $a \in P'$, the residual network G_{f_i} also contains the arc a^{-1}. Since P_i is the shortest s-t path in G_{f_i} and since the lengths of all possible s-t paths are pairwise distinct, the path $s \overset{P_i}{\rightsquigarrow} u \overset{P}{\rightsquigarrow} v \overset{P_i}{\rightsquigarrow} t$ is longer than P_i. Consequently, the path $P' = u \overset{P}{\rightsquigarrow} v$ is longer than the path $u \overset{P_i}{\rightsquigarrow} v$. This contradicts the fact that flow f_{i-1} is optimal since the arcs of path $u \overset{P_i}{\rightsquigarrow} v$ and the reverse arcs a^{-1} of the arcs a of path P' form a directed cycle C in G_{f_i} of negative costs. □

We also need to show that for each flow value the minimum-cost flow is unique with probability 1.

Lemma 5.14. *For any $\varepsilon > 0$ the probability that there exists a possible residual network with a cycle whose cost lies in $[0, \varepsilon]$ is bounded from above by $2n^{2n}\varepsilon\phi$.*

Proof. Assume that in some possible residual network G_f there exists a cycle K whose cost lies in $[0, \varepsilon]$. Then K contains two nodes u and v and consists of a u-v path P_1 and a v-u path P_2. Then P_1 and $\overleftarrow{P_2}$ are two distinct u-v paths. Since K has costs in $[0, \varepsilon]$, the costs of P_1 and $\overleftarrow{P_2}$ differ by at most ε. Now an analogous argument as in the proof of Lemma 5.6 concludes the proof. □

We can assume that the following property holds because it holds with a probability of 1. This follows from Lemma 5.14 by considering the limit for $\varepsilon \to 0$.

Property 5.15. *There does not exist a residual network that contains a cycle of cost* 0.

With Property 5.15 we can show that the minimum-cost flow is unique for each value.

Lemma 5.16. *For each value $B \in \mathbb{R}_{\geq 0}$ there either exists no flow f with $|f| = B$ or there exists a unique minimum-cost flow f with $|f| = B$.*

Proof. Assume that there exists a value $B \in \mathbb{R}_{\geq 0}$ and two distinct minimum-cost flows f and f' with $|f| = |f'| = B$. Let $E_\Delta := \{e \in E \mid f(e) \neq f'(e)\}$ be the set of edges on which f and f' differ. We show in the following that the set E_Δ contains at least one undirected cycle K. Since f and f' are distinct flows, the set E_Δ cannot be empty. For $v \in V$, let us denote by $f_-(v) = \sum_{e=(u,v)\in E} f(e)$ the flow entering v and by $f_+(v) = \sum_{e=(v,w)\in E} f(e)$ the flow going out of v ($f'_-(v)$ and $f'_+(v)$ are defined analogously). Flow conservation and $|f| = |f'|$ imply $f_-(v) - f'_-(v) = f_+(v) - f'_+(v)$ for all $v \in V$. Now let us assume E_Δ does not contain an undirected cycle. In this case there must exist a vertex $v \in V$ with exactly one incident edge in E_Δ. We will show that this cannot happen.

Assume $f_-(v) - f'_-(v) \neq 0$ for some $v \in V$. Then the flows f and f' differ on at least one edge $e = (u, v) \in E$. Since this case implies $f_+(v) - f'_+(v) \neq 0$, they also differ on at least one edge $e' = (v, w) \in E$ and both these edges belong to E_Δ. It remains to consider nodes $v \in V$ with $f_-(v) - f'_-(v) = f_+(v) - f'_+(v) = 0$ and at least one incident edge in E_Δ. For such a node v there exists an edge $e = (u, v) \in E$ (or $e = (v, w) \in E$) with $f(e) \neq f'(e)$. It follows $\sum_{e'=(u',v)\in E, e'\neq e} f(e') - f'(e') \neq 0$ (or $\sum_{e'=(v,w')\in E, e'\neq e} f(e') - f'(e') \neq 0$) which implies that there exists another edge $e' = (u', v) \neq e$ (or $e = (v, w') \neq e$) with $f(e') \neq f'(e')$. Hence every node $v \in V$ has either no incident edge from E_Δ or at least 2. This implies that E_Δ must contain an undirected cycle K.

For the flow $f'' = \frac{1}{2}f + \frac{1}{2}f'$, which has the same costs as f and f' and is hence a minimum-cost flow with $|f''| = B$ as well, we have $f''(e) \in (0, u(e))$ for all $e \in E_\Delta$. The flow f'' can therefore be augmented in both directions along K. Due to Property 5.15, augmenting f'' in one of the two directions along K will result in a better flow. This is a contradiction. □

5.2.4 Proof of Lemma 5.7

Lemma 5.17. *Let $d \geq 0$, let $P \in \mathcal{P}(G)$ be the first path encountered by the SSP algorithm with $c(P) > d$, and let $f \in \mathsf{F}$ denote the flow that induces P (i.e., P is a shortest path in the residual network G_f). Additionally, let e be an empty arc on P and let e_0 be the edge in G that corresponds to e. Now change the cost of e_0 to $c'(e_0) = 1$ if $e_0 = e$ and to $c'(e_0) = 0$ if $e_0 = e^{-1}$. In any case, the cost of arc e increases. We denote the resulting flow network by G' (i.e., G and G' coincide except for the cost of edge e_0). Then $f \in \mathsf{F}(G')$ and f is also on the network G' the first flow encountered by the SSP algorithm whose induced path P' has length strictly larger than d.*

Additionally let, $Q \in \mathcal{P}(G)$ and $Q' \in \mathcal{P}(G')$ denote the paths that the SSP algorithm encounters directly before the paths P and P', respectively. If Q and Q' exist, then $c(Q') \leq c(Q)$.

Proof. Let $\mathscr{C}$ and $\mathscr{C}'$ be the cost functions of the original network G and the modified network G', respectively. Both functions are of the form described in Lemma 5.11. We analyze the cases $e_0 = e$ and $e_0 = e^{-1}$ separately. For both cases, we introduce a function $\mathscr{C}''$ with $\mathscr{C}'' \geq \mathscr{C}$ and $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$.

We start with analyzing the case $e_0 = e$. In this case, we set $\mathscr{C}'' = \mathscr{C}'$ and observe that by increasing the cost of edge e_0 to 1 the cost of no flow can decrease. Hence, $\mathscr{C}'' \geq \mathscr{C}$. Since flow f does not use arc e, its cost remains unchanged, i.e., $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$.

If $e_0 = e^{-1}$, then we set $\mathscr{C}'' = \mathscr{C}' + \Delta(e_0)$ for $\Delta(e_0) = u(e_0) \cdot c(e_0)$. This function is also piecewise linear and has the same breakpoints and slopes as $\mathscr{C}'$. Since the flow on edge e_0 cannot exceed the capacity $u(e_0)$ of edge e_0 and since the cost on that edge has been reduced by $c(e_0)$ in G' compared to G, the cost of each flow is reduced by at most $\Delta(e_0)$ in G'. Furthermore, this gain is only achieved for flows that entirely use edge e_0 like f does. Hence, $\mathscr{C}'' \geq \mathscr{C}$ and $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$.

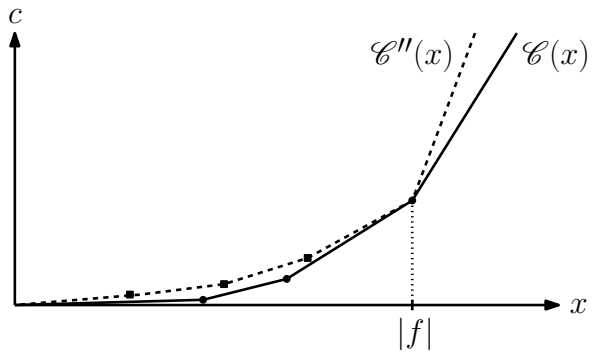


Figure 5.5: Cost function $\mathscr{C}$ and function $\mathscr{C}''$.

Due to $\mathscr{C}'' \geq \mathscr{C}$, $\mathscr{C}''(|f|) = \mathscr{C}(|f|)$, and the form of both functions, the left-hand derivative of $\mathscr{C}''$ at $|f|$ is at most the left-hand derivative of $\mathscr{C}$ at $|f|$. Similarly, the right-hand derivative of $\mathscr{C}''$ at $|f|$ is at least the right-hand derivative of $\mathscr{C}$ at $|f|$ (see Figure 5.5). Since $|f|$ is a breakpoint of $\mathscr{C}$, this implies that $|f|$ is also a breakpoint

of $\mathscr{C}''$. These properties carry over to $\mathscr{C}'$ (which either equals $\mathscr{C}''$ or is a shifted copy of $\mathscr{C}''$). Hence, Lemma 5.16 implies $f \in \mathsf{F}(G')$.

By the choice of f, the slope of $\mathscr{C}$ to the left of $|f|$ is at most d while the slope of $\mathscr{C}$ to the right of $|f|$ is larger than d. By the previous discussion about the derivatives of $\mathscr{C}$ and $\mathscr{C}'$, the same is true for $\mathscr{C}'$. This implies that f is the first flow in $\mathsf{F}(G')$ whose induced path has length strictly larger than d. Since the left-hand derivative of $\mathscr{C}'$ at $|f|$ is at most the left-hand derivative of $\mathscr{C}$ at $|f|$, it follows that $c(Q') \leq c(Q)$. □

Lemma 5.17 suggests that we can identify the first path in $\mathcal{P}$ that is longer than d without any information about the value of $c(e)$, where e is some empty arc on P. We formalize this in the following algorithm Reconstruct, which gets as input an empty arc e of P and a threshold d. The crucial fact that we will later exploit is that for this reconstruction the cost $c(e_0)$ of edge e_0 does not have to be known. (Note that we need Reconstruct only for the analysis.)

Reconstruct(e, d).

1: let e_0 be the edge that corresponds to arc e in the original network G
2: change the cost of edge e_0 to $c'(e_0) = 1$ if e is a forward arc or to $c'(e_0) = 0$ if e is a backward arc
3: start running the SSP algorithm on the modified network G'
4: stop when the length of the shortest s-t path in the residual network of the current flow f' exceeds d
5: output the shortest s-t-path in $G_{f'}$ that uses arc e (if such a path exists)

Corollary 5.18. *Let $P \in \mathcal{P}$, let e be an empty arc on P, and let $d \in \big[c(Q), c(P)\big)$, where Q denotes the path that the SSP algorithm encounters directly before P. If no such path Q exists, let $c(Q) = 0$. Then* Reconstruct(e, d) *outputs path P.*

Proof. Let $f \in \mathsf{F}$ denote the flow that induces P. By Lemma 5.17, we obtain that $f \in \mathsf{F}(G')$ and that f is the first flow encountered by the SSP algorithm whose induced path has length greater than d. This implies that Reconstruct(e, d) does not stop before encountering flow f and stops once it encounters f. Since P is the shortest path in G_f and contains arc e, Reconstruct(e, d) outputs P. □

Proof of Lemma 5.7. Let P^* denote the first path encountered by the SSP algorithm whose length is larger than d. If no such path exists, then let $P^* = \perp$. Then the event that there exists at least one path $P \in \mathcal{P}$ with $c(P) \in (d, d + \varepsilon]$ is equivalent to the event that $c(P^*) \in (d, d + \varepsilon]$. We denote by P_e the path returned by Reconstruct(e, d). Let $E^{-1} = \{e^{-1} \mid e \in E\}$. By Lemma 5.13, the path P^* contains an empty arc e^*. By Corollary 5.18, $P_{e^*} = P^*$ and hence,

$$\begin{aligned}\mathbf{Pr}\left[\exists P \in \mathcal{P} \mid c(P) \in (d, d+\varepsilon]\right] &= \mathbf{Pr}\left[c(P^*) \in (d, d+\varepsilon]\right] \\ &= \mathbf{Pr}\left[c(P_{e^*}) \in (d, d+\varepsilon]\right] \\ &\leq \mathbf{Pr}\left[\exists e \in E \cup E^{-1} \mid c(P_e) \in (d, d+\varepsilon]\right]\end{aligned}$$

$$\leq \sum_{e \in E \cup E^{-1}} \mathbf{Pr}\left[c(P_e) \in (d, d+\varepsilon]\right]. \tag{5.3}$$

Hence, it remains to analyze the probability of the event $c(P_e) \in (d, d+\varepsilon]$ for $e \in E \cup E^{-1}$. For this we use the principle of deferred decisions. It is crucial that P_e is independent of $c(e_0)$, where $e_0 \in E$ denotes the edge corresponding to the arc e. Hence, if all costs $c(e')$ with $e' \neq e_0$ are fixed, the cost of P_e can be written as either $\kappa + c(e_0)$ (if e is a forward arc) or $\kappa - c(e_0)$ (if e is a backward arc) for some constant κ that depends on the fixed values of the $c(e')$ with $e' \neq e_0$. Since $c(e_0)$ is a random variable whose density is bounded from above by ϕ, we obtain $\mathbf{Pr}\left[c(P_e) \in (d, d+\varepsilon]\right] \leq \varepsilon\phi$. Together with (5.3) this implies

$$\mathbf{Pr}\left[\exists P \in \mathcal{P} \mid c(P) \in (d, d+\varepsilon]\right] \leq 2m\phi\varepsilon. \qquad \square$$

Chapter 6

The 2-Opt Algorithm for the TSP

An instance of the *traveling salesman problem (TSP)* consists of a set $V = \{v_1, \ldots, v_n\}$ of *vertices* (depending on the context, synonymously referred to as *points*) and a symmetric *distance function* $\text{dist}\colon V \times V \to \mathbb{R}_{\geq 0}$ that assigns to each pair $\{v_i, v_j\}$ of distinct vertices a distance $\text{dist}(v_i, v_j) = \text{dist}(v_j, v_i)$. The goal is to find a Hamiltonian cycle (i.e., a cycle that visits every vertex exactly once) of minimum length. We also use the term *tour* to denote a Hamiltonian cycle.

A pair (V, dist) of a nonempty set V and a function $\text{dist}\colon V \times V \to \mathbb{R}_{\geq 0}$ is called a *metric space* if for all $x, y, z \in V$ the following properties are satisfied:

(a) $\text{dist}(x, y) = 0$ if and only if $x = y$ *(reflexivity)*,

(b) $\text{dist}(x, y) = \text{dist}(y, x)$ *(symmetry)*,

(c) $\text{dist}(x, z) \leq \text{dist}(x, y) + \text{dist}(y, z)$ *(triangle inequality)*.

If (V, dist) is a metric space, then dist is called a *metric on* V. A TSP instance with vertices V and distance function dist is called *metric TSP instance* if (V, dist) is a metric space.

A well-known class of metrics on $\mathbb{R}^d$ is the class of L_p *metrics*. For $p \in \mathbb{N}$, the distance $\text{dist}_p(x, y)$ of two points $x \in \mathbb{R}^d$ and $y \in \mathbb{R}^d$ with respect to the L_p metric is given by $\text{dist}_p(x, y) = \sqrt[p]{|x_1 - y_1|^p + \cdots + |x_d - y_d|^p}$. The L_1 metric is often called *Manhattan metric*, and the L_2 metric is well-known as *Euclidean metric*. For $p \to \infty$, the L_p metric converges to the L_∞ metric defined by the distance function $\text{dist}_\infty(x, y) = \max\{|x_1 - y_1|, \ldots, |x_d - y_d|\}$. A TSP instance (V, dist) with $V \subseteq \mathbb{R}^d$ in which dist equals dist_p restricted to V is called an L_p *instance*. We also use the terms *Manhattan instance* and *Euclidean instance* to denote L_1 and L_2 instances, respectively. Furthermore, if p is clear from context, we write dist instead of dist_p.

6.1 Overview of Results

Despite many theoretical analyses and experimental evaluations of the TSP, there is still a considerable gap between the theoretical results and the experimental obser-

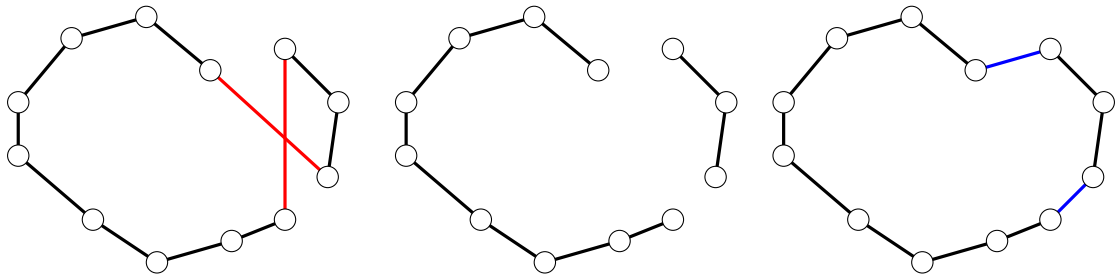


Figure 6.1: Example of a 2-change in which the red edges are exchanged with the blue edges.

vations. One important special case is the *Euclidean TSP* in which the vertices are points in $\mathbb{R}^d$, for some $d \in \mathbb{N}$, and the distances are measured according to the Euclidean metric. This special case is known to be NP-hard in the strong sense [27], but for any constant dimension d it admits a polynomial time approximation scheme (PTAS), shown independently in 1996 by Arora [1] and Mitchell [23]. These approximation schemes are based on dynamic programming. However, the most successful algorithms on practical instances rely on the principle of local search and very little is known about their complexity.

The *2-Opt* algorithm is probably the simplest local search heuristic for the TSP. 2-Opt starts with an arbitrary initial tour and incrementally improves this tour by successive improvements that exchange two of the edges in the tour with two other edges. More precisely, in each *improving step* the 2-Opt algorithm selects two edges $\{u_1, u_2\}$ and $\{v_1, v_2\}$ from the tour such that u_1, u_2, v_1, v_2 are distinct and appear in this order in the tour, and it replaces these edges by the edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$, provided that this change decreases the length of the tour (see Figure 6.1). The algorithm terminates in a local optimum in which no further improving step is possible. We use the term *2-change* to denote a local improvement made by 2-Opt. This simple heuristic performs amazingly well on "real-life" Euclidean instances like, e.g., the ones in the well-known TSPLIB [28]. Usually the 2-Opt heuristic needs a clearly subquadratic number of improving steps until it reaches a local optimum and the computed solution is within a few percentage points of the global optimum [14].

There are numerous experimental studies on the performance of 2-Opt. However, the theoretical knowledge about this heuristic is still very limited. Let us first discuss the number of local improvement steps made by 2-Opt before it finds a locally optimal solution. When talking about the number of local improvements, it is convenient to consider the *state graph*. The vertices in this graph correspond to the possible tours and an arc from a vertex v to a vertex u is contained if u is obtained from v by performing an improving 2-change. On the positive side, van Leeuwen and Schoone consider a 2-Opt variant for the Euclidean plane in which only steps are allowed that remove a crossing from the tour. Such steps can introduce new crossings, but van Leeuwen and Schoone [31] show that after $O(n^3)$ steps, 2-Opt finds a tour without any crossing. On the negative side, Lueker [21] constructs TSP instances whose state graphs contain exponentially long paths. Hence, 2-Opt can take an exponential number of steps before it finds a locally optimal solution. This result is generalized to k-Opt, for

arbitrary $k \geq 2$, by Chandra, Karloff, and Tovey [11]. These negative results, however, use arbitrary graphs that cannot be embedded into low-dimensional Euclidean space. Hence, they leave open the question as to whether it is possible to construct Euclidean TSP instances on which 2-Opt can take an exponential number of steps. This question is resolved by Englert et al. [12] by constructing such instances in the Euclidean plane. In chip design applications, often TSP instances arise in which the distances are measured according to the Manhattan metric. Also for this metric and for every other L_p metric, Englert et al. construct instances with exponentially long paths in the 2-Opt state graph.

Theorem 6.1 ([12])**.** *For every $p \in \mathbb{N} \cup \{\infty\}$ and $n \in \mathbb{N}$, there is a two-dimensional TSP instance with $16n$ vertices in which the distances are measured according to the L_p metric and whose state graph contains a path of length $2^{n+4} - 22$.*

For Euclidean instances in which n points are placed independently uniformly at random in the unit square, Kern [17] shows that the length of the longest path in the state graph is bounded by $O(n^{16})$ with probability at least $1 - c/n$ for some constant c. Chandra, Karloff, and Tovey [11] improve this result by bounding the *expected* length of the longest path in the state graph by $O(n^{10} \log n)$. That is, independent of the initial tour and the choice of the local improvements, the expected number of 2-changes is bounded by $O(n^{10} \log n)$. For instances in which n points are placed uniformly at random in the unit square and the distances are measured according to the Manhattan metric, Chandra, Karloff, and Tovey show that the expected length of the longest path in the state graph is bounded by $O(n^6 \log n)$.

Englert et al. consider a more general probabilistic input model, which is a canonical extension of ϕ-perturbed numbers to higher dimensions, and improve the previously known bounds. The probabilistic model underlying their analysis allows different vertices to be placed independently according to different continuous probability distributions in the unit hypercube $[0,1]^d$, for some constant *dimension* $d \geq 2$. The distribution of a vertex v_i is defined by a density function $f_i \colon [0,1]^d \to [0,\phi]$ for some given $\phi \geq 1$. The upper bounds depend on the number n of vertices and the upper bound ϕ on the density. We denote instances created by this input model as *ϕ-perturbed Euclidean* or *Manhattan instances*, depending on the underlying metric. The parameter ϕ can be seen as a parameter specifying how close the analysis is to a worst case analysis because the larger ϕ, the better worst-case instances can be approximated by the distributions. For $\phi = 1$ and $d = 2$, every point has a uniform distribution over the unit square, and hence the input model equals the uniform model analyzed before.

Englert et al. also consider a model in which an arbitrary graph $G = (V, E)$ is given and for each edge $e \in E$, a probability distribution according to which the edge length $\mathrm{dist}(e)$ is chosen independently of the other edge lengths. Again, we restrict the choice of distributions to distributions that can be represented by density functions $f_e \colon [0,1] \to [0,\phi]$ with maximal density at most ϕ for a given $\phi \geq 1$. We denote inputs created by this input model as *ϕ-perturbed graphs*. Observe that in this input model only the distances are perturbed whereas the graph structure is not changed by the randomization. This can be useful if one wants to explicitly prohibit certain

edges. However, if the graph G is not complete, one has to initialize 2-Opt with a Hamiltonian cycle to start with.

Englert et al. prove the following theorem about the expected length of the longest path in the 2-Opt state graph for the probabilistic input models discussed above. It is assumed that the dimension $d \geq 2$ is an arbitrary constant.

Theorem 6.2 ([12])**.** *The expected length of the longest path in the 2-Opt state graph*

a) is $O(n^4 \cdot \phi)$ for ϕ-perturbed Manhattan instances with n points,

b) is $O(n^{4+1/3} \cdot \log(n\phi) \cdot \phi^{8/3})$ for ϕ-perturbed Euclidean instances with n points,

c) is $m^{1+o(1)} \cdot n \cdot \phi$ for ϕ-perturbed graphs with n vertices and m edges.

Similar to the running time, the good approximation ratios obtained by 2-Opt on practical instances cannot be explained by a worst-case analysis. In fact, there are quite negative results on the worst-case behavior of 2-Opt. For example, Chandra, Karloff, and Tovey [11] show that there are Euclidean instances in the plane for which 2-Opt has local optima whose costs are $\Omega\left(\frac{\log n}{\log \log n}\right)$ times larger than the optimal costs. However, the same authors also show that the expected approximation ratio of the worst local optimum for instances with n points drawn uniformly at random from the unit square is bounded from above by a constant. Their result can be generalized to the input model in which different points can have different distributions with bounded density ϕ and to all L_p metrics.

Theorem 6.3 ([12])**.** *Let $p \in \mathbb{N} \cup \{\infty\}$. For ϕ-perturbed L_p instances, the expected approximation ratio of the worst tour that is locally optimal for 2-Opt is $O(\sqrt[d]{\phi})$.*

6.2 Polynomial Bound for ϕ-Perturbed Graphs

Is this section, we present a glimpse into the proof of Theorem 6.2. The complete proof is too technical to be presented in the lecture in detail, but we will outline the main ideas by proving a weaker version of part c) of the theorem.

Theorem 6.4. *For any ϕ-perturbed graph with n vertices and m edges, the expected length of the longest path in the 2-Opt state graph is $O(n^2m^2 \log(n) \cdot \phi)$.*

Proof. How can we prove an upper bound on the (expected) number of steps made by 2-Opt? For this, we use the length of the current tour as a *potential function*. As all edge lengths lie in the interval $[0, 1]$, any tour (in particular the one 2-Opt starts with) has length at most n. If we know that every 2-change decreases the length of the current tour by at least $\Delta > 0$, then we can bound the number of 2-changes that can be made before reaching a local optimum from above by n/Δ because after that many steps the length of the tour must have decreased to zero. As it cannot get negative, no further local improvement can be possible.

Hence, if we show that the smallest possible improvement Δ is not too small, then it follows that 2-Opt cannot make too many steps. Unfortunately, in the worst case one

can easily construct a set of points that allows a 2-change that decreases the length of the tour only by an arbitrarily small amount. Our goal is to show that on ϕ-perturbed graphs, Δ does not become too small with high probability.

Let us first consider a fixed 2-change in which the edges e_1 and e_2 are exchanged with the edges e_3 and e_4. This 2-change decreases the length of the tour by

$$\Delta(e_1, e_2, e_3, e_4) = \operatorname{dist}(e_1) + \operatorname{dist}(e_2) - \operatorname{dist}(e_3) - \operatorname{dist}(e_4). \tag{6.1}$$

We define Δ as the smallest possible improvement made by any improving 2-change:

$$\Delta = \min_{\substack{e_1,e_2,e_3,e_4 \\ \Delta(e_1,e_2,e_3,e_4)>0}} \Delta(e_1, e_2, e_3, e_4),$$

where the minimum is taken over all tuples $(e_1, e_2, e_3, e_4) \in E^4$ for which e_1, e_3, e_2, e_4 is a 4-cycle in G because only these tuples could possibly form a 2-change.

The following lemma, which is proven below, is one crucial ingredient of the proof.

Lemma 6.5. *For any $\varepsilon > 0$,*

$$\mathbf{Pr}\left[\Delta \le \varepsilon\right] \le m^2 \varepsilon \phi.$$

With the help of this lemma we can prove the theorem. We have argued above that the number of steps that 2-Opt can make is bounded from above by n/Δ. Let T denote the maximal number of steps 2-Opt can make, i.e., the length of the longest path in the state graph. This number can only be greater than or equal to t if $n/\Delta \ge t$, which is equivalent to $\Delta \le n/t$. Hence,

$$\mathbf{Pr}\left[T \ge t\right] \le \mathbf{Pr}\left[\Delta \le \frac{n}{t}\right] \le \frac{nm^2\phi}{t}.$$

One important observation is that T is always bounded from above by $n!$ because this is an upper bound on the number of possible tours. Hence, we obtain the following bound for the expected value of T:

$$\mathbf{E}\left[T\right] = \sum_{t=1}^{n!} \mathbf{Pr}\left[T \ge t\right] \le \sum_{t=1}^{n!} \frac{nm^2\phi}{t} = nm^2\phi \cdot \sum_{t=1}^{n!} \frac{1}{t}.$$

Using that

$$\sum_{t=1}^{n!} \frac{1}{t} = O(\log(n!)) = O(n \log(n))$$

yields

$$\mathbf{E}\left[T\right] = O(nm^2\phi \log(n!)) = O(n^2 m^2 \log(n) \cdot \phi). \qquad \square$$

To complete the proof of the Theorem, we have to prove Lemma 6.5.

Proof of Lemma 6.5. Again we first consider a fixed 2-change in which the edges e_1 and e_2 are exchanged with the edges e_3 and e_4. We would like to bound the probability that this fixed 2-change is improving, but yields an improvement of at most ε, for some $\varepsilon > 0$. That is, we want to bound the following probability from above:

$$\mathbf{Pr}\left[\Delta(e_1, e_2, e_3, e_4) \in (0, \varepsilon]\right] = \mathbf{Pr}\left[\text{dist}(e_1) + \text{dist}(e_2) - \text{dist}(e_3) - \text{dist}(e_4) \in (0, \varepsilon]\right].$$

We use the principle of deferred decisions and assume that the lengths $\text{dist}(e_2)$, $\text{dist}(e_3)$, and $\text{dist}(e_4)$ have already been fixed arbitrarily. Then the event $\Delta(e_1, e_2, e_3, e_4) \in (0, \varepsilon]$ is equivalent to the event that

$$\text{dist}(e_1) \in (\kappa, \kappa + \varepsilon],$$

where $\kappa = \text{dist}(e_4) + \text{dist}(e_3) - \text{dist}(e_2)$ is some fixed value. As $\text{dist}(e_1)$ is a random variable whose density is bounded from above by ϕ, the probability that $\text{dist}(e_1)$ assumes a value in a fixed interval of length ε is at most $\varepsilon\phi$.

This bound makes only a statement about the improvement made by a particular step in which the edges e_1 and e_2 are exchanged with the edges e_3 and e_4, but we would like to make a statement about every possible 2-change. For this we apply a union bound over all possible 2-changes. There are $\binom{m}{2} < \frac{m^2}{2}$ choices for the set $\{e_1, e_2\}$ and, once this set is fixed, there are two choices for the set $\{e_3, e_4\}$ because e_1, e_3, e_2, e_4 has to be a 4-cycle. Hence, the total number of different 2-changes is bounded from above by m^2, which yields

$$\mathbf{Pr}\left[\Delta \in (0, \varepsilon]\right] \leq \mathbf{Pr}\left[\exists e_1, e_2, e_3, e_4 : \Delta(e_1, e_2, e_3, e_4) \in (0, \varepsilon]\right] \leq m^2 \varepsilon \phi,$$

where the existential quantifier in the second probability is over all $(e_1, e_2, e_3, e_4) \in E^4$ for which e_1, e_3, e_2, e_4 is a 4-cycle in G. This concludes the proof of the lemma. □

6.3 Improved Analysis

The bound in Theorem 6.4 is only based on the smallest improvement Δ made by any of the 2-Opt steps. Intuitively, this is too pessimistic because most of the steps performed by 2-Opt yield a larger improvement than Δ. In particular, two consecutive steps yield an improvement of at least Δ plus the improvement Δ' of the second smallest step. This observation alone, however, does not suffice to improve the bound substantially.

To obtain a significantly better bound, we show that it is possible to regroup most of the 2-changes in any sufficiently long sequence to pairs such that each pair of 2-changes is *linked* by an edge, i.e., one edge added to the tour in the first 2-change is removed from the tour in the second 2-change. Then we analyze the smallest improvement made by any pair of linked 2-changes. Obviously, this improvement is at least $\Delta + \Delta'$ but one can hope that it is much larger because it is unlikely that the 2-change that yields the smallest improvement and the 2-change that yields the second smallest improvement form a pair of linked steps. It can be shown that this is indeed the case. This result is then used to prove another weaker version of part c) of Theorem 6.2 that improves upon Theorem 6.4.

Theorem 6.6. *For any ϕ-perturbed graph with n vertices and m edges, the expected length of the longest path in the 2-Opt state graph is $O(nm^{3/2}\phi)$.*

Construction of Pairs of Linked 2-Changes

Consider an arbitrary sequence of t consecutive 2-changes. The following lemma guarantees that the number of disjoint linked pairs of 2-changes in every such sequence increases linearly with the length t.

Lemma 6.7. *In every sequence of t consecutive 2-changes performed by 2-Opt, we can find at least $(2t - n)/7$ disjoint pairs of 2-changes that are linked by an edge, i.e., pairs where there exists an edge added to the tour in the first 2-change of the pair and removed from the tour in the second 2-change of the pair.*

Proof. Let $S_1, \ldots, S_t$ denote an arbitrary sequence of consecutive 2-changes. The sequence is processed step by step and a list $\mathcal{L}$ of linked pairs of 2-changes is created. The pairs in $\mathcal{L}$ are not necessarily disjoint. Hence, after the list has been created, pairs have to be removed from the list until there are no non-disjoint pairs left. Assume that the 2-changes $S_1, \ldots, S_{i-1}$ have already been processed and that now 2-change S_i has to be processed. Assume further that in step S_i the edges e_1 and e_2 are exchanged with the edges e_3 and e_4. Let j denote the smallest index with $j > i$ such that edge e_3 is removed from the tour in step S_j if such a step exists. In this case, the pair (S_i, S_j) is added to the list $\mathcal{L}$. Analogously, let j' denote the smallest index with $j' > i$ such that edge e_4 is removed from the tour in step $S_{j'}$ if such a step exists. In this case, also the pair $(S_i, S_{j'})$ is added to the list $\mathcal{L}$.

After the sequence has been processed completely, each pair in $\mathcal{L}$ is linked by an edge but we still have to identify a subset $\mathcal{L}'$ of $\mathcal{L}$ consisting only of pairwise disjoint pairs. This subset is constructed in a greedy fashion. We process the list $\mathcal{L}$ step by step, starting with an empty list $\mathcal{L}'$. For each pair in $\mathcal{L}$, we check whether it is disjoint from all pairs that have already been inserted into $\mathcal{L}'$ or not. In the former case, the current pair is inserted into $\mathcal{L}'$. This way, we obtain a list $\mathcal{L}'$ of disjoint pairs such that each pair is linked by an edge. The number of pairs in $\mathcal{L}$ is at least $2t - n$ because each of the t steps gives rise to 2 pairs, unless an edge is added to the tour that is never removed again. The tour C obtained after the 2-changes $S_1, \ldots, S_t$ contains exactly n edges. For every edge $e \in C$, only the last step in which e enters the tour (if such a step exists) does not create a pair of linked 2-changes involving e.

Each 2-change occurs in at most 4 different pairs in $\mathcal{L}$. In order to see this, consider a 2-change S in which the edges e_1 and e_2 are exchanged with the edges e_3 and e_4. Then L contains the following pairs involving S (if they exist): (S, S') where S' is either the first step after S in which e_3 gets removed or the first step after S in which e_4 gets removed, and (S', S) where S' is either the last step before S in which e_1 enters the tour or the last step before S in which e_2 enters the tour. With similar reasoning, one can argue that each pair in $\mathcal{L}$ is non-disjoint from at most 6 other pairs in $\mathcal{L}$. This implies that $\mathcal{L}$ contains at most 7 times as many pairs as $\mathcal{L}'$, which concludes the proof. □

Analysis of Pairs of Linked 2-Changes

The following lemma gives a bound on the probability that there exists a linked pair of 2-changes in which both steps are small improvements.

Lemma 6.8. *In any ϕ-perturbed graph with n vertices and m edges, the probability that there exists a pair of linked 2-changes that are both improvements by at most ε is $O(m^3\phi^2\varepsilon^2)$.*

Proof. First consider a fixed pair of linked 2-changes. Assume that in the first 2-change the edges e_1 and e_2 are replaced by the edges e_3 and e_4 and that in the second 2-change the edges e_4 and e_5 are replaced by the edges e_6 and e_7. The sets $\{e_1, e_2, e_3\}$ and $\{e_5, e_6, e_7\}$ are not necessarily disjoint. They are, however, distinct because otherwise the second 2-change would be exactly the reverse of the first 2-change (i.e., the edges e_3 and e_4 are replaced by the edges e_1 and e_2). In this case, at least one of the 2-changes is not an improvement.

Let us assume without loss of generality that $e_6 \notin \{e_1, e_2, e_3\}$ and $e_1 \notin \{e_5, e_6, e_7\}$. Our goal is to give an upper bound on the probability that both

$$\Delta(e_1, e_2, e_3, e_4) := \operatorname{dist}(e_1) + \operatorname{dist}(e_2) - \operatorname{dist}(e_3) - \operatorname{dist}(e_4) \in (0, \varepsilon]$$

and

$$\Delta(e_4, e_5, e_6, e_7) := \operatorname{dist}(e_4) + \operatorname{dist}(e_5) - \operatorname{dist}(e_6) - \operatorname{dist}(e_7) \in (0, \varepsilon].$$

We use the principle of deferred decisions and assume that the lengths of the edges e_2, e_3, e_4, e_5, and e_7 are already revealed. For any fixed realization of these edge lengths, the events

$$\Delta(e_1, e_2, e_3, e_4) \in (0, \varepsilon] \quad \text{and} \quad \Delta(e_4, e_5, e_6, e_7) \in (0, \varepsilon]$$

can be written as

$$\operatorname{dist}(e_1) \in (\kappa_1, \kappa_1 + \varepsilon] \quad \text{and} \quad \operatorname{dist}(e_6) \in [\kappa_2 - \varepsilon, \kappa_2),$$

where $\kappa_1 = \operatorname{dist}(e_4) + \operatorname{dist}(e_3) - \operatorname{dist}(e_2)$ and $\kappa_2 = \operatorname{dist}(e_4) + \operatorname{dist}(e_5) - \operatorname{dist}(e_7)$ are constants depending on the realization of the edges e_2, e_3, e_4, e_5, and e_7. Hence, for any fixed realization these events are independent. Each of them occurs with a probability of at most $\phi\varepsilon$ because $\operatorname{dist}(e_1)$ and $\operatorname{dist}(e_6)$ are random variables whose densities are bounded from above by ϕ. In summary, for any linked pair of 2-changes the probability that both steps are improvements by at most ε is bounded from above by $\phi^2\varepsilon^2$.

To conclude the proof it suffices to apply a union bound over all possible pairs of linked 2-changes. As argued in the proof of Theorem 6.4, there are at most m^2 choices for the first 2-change. Once this 2-change is fixed, there are only two choices for the edge that links the first and the second 2-change. There are at most m choices for the other edge removed from the tour in the second step. Once the edges are fixed that are removed from the tour in the second step, there are only two possibilities for the edges added to the tour in that step. In total, there are only $O(m^3)$ different pairs of linked 2-changes. This completes the proof. □

Expected Number of 2-Changes

Based on Lemmas 6.7 and 6.8, we are now able to prove Theorem 6.6.

Proof of Theorem 6.6. Let T denote the random variable that describes the length of the longest path in the state graph. If $T \geq t$, then there must exist a sequence $S_1, \ldots, S_t$ of t consecutive 2-changes in the state graph. We start by identifying a set of disjoint linked pairs of 2-changes in this sequence. Due to Lemma 6.7, we know that we can find at least $z = (2t-n)/7$ such pairs. Let $\Delta^\star$ denote the smallest improvement made by any pair of linked 2-changes. If $T \geq t$, then $\Delta^\star \leq n/z$ as the initial tour has length at most n and every pair of linked 2-changes decreases the length of the tour by at least $\Delta^\star$. For $t \geq n$, we have $z = (2t-n)/7 \geq t/7$ and hence due to Lemma 6.8,

$$\mathbf{Pr}\left[T \geq t\right] \leq \mathbf{Pr}\left[\Delta^\star \leq \frac{n}{z}\right] \leq \mathbf{Pr}\left[\Delta^\star \leq \frac{7n}{t}\right] = O\left(\frac{m^3 n^2 \phi^2}{t^2}\right).$$

Using the fact that probabilities are bounded from above by one, we obtain, for some constant κ,

$$\mathbf{Pr}\left[T \geq t\right] = \min\left\{\frac{\kappa m^3 n^2 \phi^2}{t^2}, 1\right\}.$$

Since T cannot exceed $n!$, this implies the following bound on the expected number of 2-changes:

$$\begin{aligned}
\mathbf{E}\left[T\right] &\leq \sum_{t=1}^{n!} \mathbf{Pr}\left[T \geq t\right] \\
&\leq \sum_{t=1}^{n!} \min\left\{\frac{\kappa m^3 n^2 \phi^2}{t^2}, 1\right\} \\
&\leq m^{3/2} n\phi + \sum_{t=m^{3/2}n\phi}^{n!} \frac{\kappa m^3 n^2 \phi^2}{t^2} \\
&\leq m^{3/2} n\phi + \int_{m^{3/2}n\phi}^{\infty} \frac{\kappa m^3 n^2 \phi^2}{t^2}\, dt \\
&= m^{3/2} n\phi + \left[-\frac{\kappa m^3 n^2 \phi^2}{t}\right]_{m^{3/2}n\phi}^{\infty} \\
&= O(m^{3/2} n\phi).
\end{aligned}$$

This concludes the proof of the theorem. □

Part a) and b) of Theorem 6.2 can be proven with similar arguments that we used to prove Theorem 6.6. However, the calculations are much more involved because in Manhattan and Euclidean instances the lengths of different edges are not independent anymore if they share a vertex. The main idea of the proof of part c) is to analyze not only pairs of linked 2-changes but longer sequences of linked 2-changes.

Chapter 7

The k-Means Method

Clustering is a fundamental problem in computer science with applications ranging from biology to information retrieval and data compression. In a clustering problem, a set of objects, usually represented as points in a high-dimensional space $\mathbb{R}^d$, is to be partitioned such that objects in the same group share similar properties. The k-means method is a very simple heuristic for clustering. It is used to partition a finite set $X \subseteq \mathbb{R}^d$ of n d-dimensional data points into k clusters, where the number k of clusters is fixed in advance.

In k-means clustering, our goal is not only to get a clustering of the data points, but also to get a *center* c_i for each cluster C_i of the clustering $C_1, \ldots, C_k$. This center can be viewed as a representative of its cluster. We do not require the centers to be among the data points, but they can be arbitrary points in $\mathbb{R}^d$.

The goal is to find a clustering that minimizes the objective function

$$\Psi = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - c_i\|^2 .$$

Here, $\|x - c_i\|$ denotes the Euclidean distance between x and c_i. We will refer to the objective value Ψ also as *potential*.

Given the cluster centers $c_1, \ldots, c_k \in \mathbb{R}^d$, each point $x \in X$ should be assigned to the cluster C_i that is represented by its closest center c_i. On the other hand, given a clustering $C_1, \ldots, C_k$ of the data points, each center c_i should be chosen as the center of mass $\mathrm{cm}(C_i) := \frac{1}{|C_i|} \cdot \sum_{x \in C_i} x$ of C_i in order to minimize the objective function (Lemma 7.3).

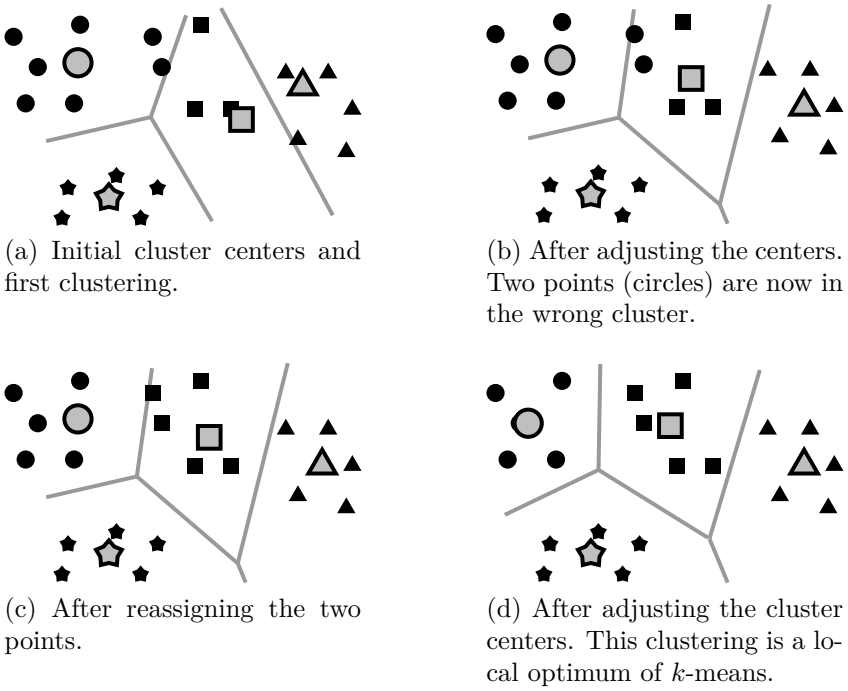
The *k-means method* (often called *k-means* for short or *Lloyd's method* because it is based on ideas by Lloyd [20]) exploits this duality between clustering and centers: Given the centers, it is clear how the clustering should be chosen. And given the clustering, we know where the centers should be. We start with some clustering and then alternatingly optimize centers and clustering until this process stabilizes and we have reached a local optimum. Algorithm 1 states this more formally. Figure 7.1 provides an example run of the k-means method in two dimensions. Given the k

centers, the Voronoi cell of a center c_i consists of all points closer to c_i than to any other center. The gray lines in Figure 7.1 are the borders between the Voronoi cells.

Algorithm 1 The k-means method.

Input: set X of n data points in $\mathbb{R}^d$, number k of clusters
Output: clustering $C_1, \ldots, C_k$ and centers $c_1, \ldots, c_k$.

1: Choose initial cluster centers $c_1, \ldots, c_k \in \mathbb{R}^d$ arbitrarily.
2: For each point $x \in X$: If c_i is the center closest to x, then assign x to C_i. (We assume a consistent tie-breaking if the closest center is not unique.)
3: For each $i \in \{1, \ldots, k\}$: Set $c_i = \mathrm{cm}(C_i)$.
4: If anything has changed in steps 2 and 3, then go back to step 2.



(a) Initial cluster centers and first clustering.

(b) After adjusting the centers. Two points (circles) are now in the wrong cluster.

(c) After reassigning the two points.

(d) After adjusting the cluster centers. This clustering is a local optimum of k-means.

Figure 7.1: An example of k-means.

In the following, an *iteration* of k-means refers to one execution of step 2 followed by step 3. A slight technical subtlety in the implementation of the algorithm is the possible event that a cluster loses all its points in step 2. There exist different strategies to deal with this case. For simplicity, we use the strategy of removing clusters that serve no points and continuing with the remaining clusters.

The k-means method is one of the most popular clustering algorithms used in scientific and industrial applications. The main reason for its popularity is its speed. This, however, is in stark contrast to its performance in theory: The worst-case running time of k-means has recently been shown to be exponential in the number k of clusters [32]. The only known upper bound for its running time is $(k^2 n)^{kd}$ [13], but this bound is far

from explaining the speed of k-means. It is solely based on the observation that the k-means method cannot visit the same clustering twice because the objective function Ψ decreases monotonically during the execution of the k-means method.

In order to explain the practical performance of k-means and to reconcile theory and practice, a smoothed analysis of k-means has been conducted. Here, an adversary specifies a set $X' \subseteq [0,1]^d$ of n points. Then each point from X' is independently perturbed by a normal distribution with mean 0 and standard deviation σ, yielding a new set X of points. We call X a *σ-smooth point set* (formally a σ-smooth point set is not a set of points but a set of random vectors). In a series of papers [3, 22, 2], it has been shown that the smoothed running time (i.e., the expected running time on σ-smooth point sets) of the k-means method is bounded from above by a polynomial in the number n of data points and $1/\sigma$, i.e., it is bounded by $\text{poly}(n, 1/\sigma)$, where the degree of the polynomial depends neither on k nor on d. As the proof of this statement is rather involved, we prove in this lecture only the following weaker version from [3]. We assume for the sake of simplicity that $2 \leq d \leq n$, $k \leq n$, and $\sigma \leq 1$ even though none of these restrictions is crucial for the following result.

Theorem 7.1 ([3])**.** *Let X be a σ-smooth point set of cardinality n. Then the expected running time of k-means on X is $O((n^k/\sigma)^c)$, where the constant c is independent of d and k.*

Theorem 7.1 shows that the smoothed running time of the k-means method is polynomially bounded in n^k and $1/\sigma$. This is already a significant improvement over the best known worst-case upper bound, which is polynomial in n^{kd} instead of n^k.

In order to prove the theorem, we follow an approach similar to the analysis of 2-Opt in the previous chapter. That is, we bound the smallest improvement Δ of the objective value Ψ in any possible iteration of the k-means method from below. Before we discuss this in more detail, we will first discuss which events lead to a decrease of Ψ. Then we analyze in Sections 7.2 and 7.3 the probability that Δ is small for iterations in which many or few points change their assignment, respectively.

7.1 Potential Drop in an Iteration of k-Means

During an iteration of the k-means method there are two possible events that can lead to a significant potential drop: either one cluster center moves significantly, or a data point is reassigned from one cluster to another and this point has a significant distance from the bisector of the clusters (the bisector is the hyperplane that bisects the two cluster centers). In the following we quantify the potential drops caused by these events.

The potential drop caused by reassigning a data point x from one cluster to another can be expressed in terms of the distance of x from the bisector of the two cluster centers and the distance between these two centers.

Lemma 7.2. *Assume that in an iteration of k-means a point $x \in X$ switches from C_i to C_j. Let c_i and c_j be the centers of these clusters and let H be their bisector. Then reassigning x decreases the potential by $2 \cdot \|c_i - c_j\| \cdot \operatorname{dist}(x, H)$.*

Proof. The potential decreases by

$$\begin{aligned}
\|c_i - x\|^2 - \|c_j - x\|^2 &= \sum_{\ell=1}^{d} \left[[(c_i)_\ell - x_\ell]^2 - [(c_j)_\ell - x_\ell]^2 \right] \\
&= \sum_{\ell=1}^{d} \left[(c_i)_\ell^2 - (c_j)_\ell^2 - 2x_\ell \left[(c_i)_\ell - (c_j)_\ell \right] \right] \\
&= \sum_{\ell=1}^{d} \left[\left[(c_i)_\ell - (c_j)_\ell \right] \left[(c_i)_\ell + (c_j)_\ell \right] - 2x_\ell \left[(c_i)_\ell - (c_j)_\ell \right] \right] \\
&= \sum_{\ell=1}^{d} \left[\left[(c_i)_\ell - (c_j)_\ell \right] \left[(c_i)_\ell + (c_j)_\ell - 2x_\ell \right] \right] \\
&= (c_i - c_j) \cdot (c_i + c_j - 2x) \\
&= (2x - c_i - c_j) \cdot (c_j - c_i).
\end{aligned}$$

Let v be the unit vector in the $c_j - c_i$ direction. Then $(2x - c_i - c_j) \cdot v = 2 \cdot \operatorname{dist}(x, H)$ because v is orthogonal to H and c_i and c_j have the same distance to H, which implies $c_i \cdot v = -c_j \cdot v$. The observation $c_j - c_i = \|c_i - c_j\| \cdot v$ completes the proof. □

The following lemma, which also follows from basic linear algebra, reveals how moving a cluster center to the center of mass decreases the potential.

Lemma 7.3 ([15])**.** *Assume that the center of a cluster C moves from c to $\operatorname{cm}(C)$ during an iteration of k-means, and let $|C|$ denote the number of points in C when the movement occurs. Then the potential decreases by $|C| \cdot \|c - \operatorname{cm}(C)\|^2$.*

Proof. It is well-known that it is possible to express the contribution of C to the current potential based on the center of C. We will see this in the following. Then, we can compute how much changing the center changes the potential.

We first recall the following calculation rule that holds for all $x, y \in \mathbb{R}^d$:

$$||x + y||^2 = (x + y) \cdot (x + y) = x \cdot x + 2 \cdot x \cdot y + y \cdot y = ||x||^2 + 2 \cdot x \cdot y + ||y||^2.$$

This allows us to conveniently rewrite the squared distance of a point $x \in C$ to an arbitrary center $z \in \mathbb{R}^d$ as $||x - z||^2 = ||x - \operatorname{cm}(C) + \operatorname{cm}(C) - z||^2 = ||x - \operatorname{cm}(C)||^2 + 2(\operatorname{cm}(C) - z) \cdot (x - \operatorname{cm}(C)) + ||\operatorname{cm}(C) - z||^2$. We would like to get rid of the middle term. Luckily, it disappears when we compute the squared distances of *all* points to z. More precisely, we observe that

$$\sum_{x \in C} (x - \operatorname{cm}(C)) = \left(\sum_{x \in C} x \right) - |C| \cdot \operatorname{cm}(C) = \left(\sum_{x \in C} x \right) - \left(\sum_{x \in C} x \right) = 0,$$

which helps us to compute C's contribution to the potential with center z as:

$$\begin{aligned}
&\sum_{x\in C} ||x-z||^2 \\
=&\sum_{x\in C} ||x-\mathrm{cm}(C)||^2 + \sum_{x\in C} 2(x-\mathrm{cm}(C))\cdot(\mathrm{cm}(C)-z) + \sum_{x\in C} ||z-\mathrm{cm}(C)||^2 \\
=&\sum_{x\in C} ||x-\mathrm{cm}(C)||^2 + 2(\mathrm{cm}(C)-z)\cdot\sum_{x\in C}(x-\mathrm{cm}(C)) + |C|\cdot||z-\mathrm{cm}(C)||^2 \\
=&\sum_{x\in C} ||x-\mathrm{cm}(C)||^2 + 2(\mathrm{cm}(C)-z)\cdot 0 + |C|\cdot||z-\mathrm{cm}(C)||^2 \\
=&\sum_{x\in C} ||x-\mathrm{cm}(C)||^2 + |C|\cdot||z-\mathrm{cm}(C)||^2.
\end{aligned}$$

We see that $\sum_{x\in C} ||x-\mathrm{cm}(C)||^2$ appears in the potential independently of the center. For center $\mathrm{cm}(C)$, the additional term is zero. For center c, the additional term is $|C|\cdot||c-\mathrm{cm}(C)||^2$. So changing the center from c to $\mathrm{cm}(C)$ decreases the potential by exactly $|C|\cdot||c-\mathrm{cm}(C)||^2$. □

7.2 Iterations with Large Cluster Changes

In this section, we consider iterations in which one cluster C gains or loses in total at least $2kd$ points. In this case, there must exist another cluster C' such that C and C' exchange at least $2d+1$ points. We show that with high probability at least one of these points is not too close to the bisector of C and C'. Together with Lemma 7.2 this implies that the potential decreases significantly.

Definition 7.4. *Let $X \subseteq \mathbb{R}^d$ be a set of points. We say that X is* δ-separated *if for any hyperplane H there are at most $2d$ points in X within distance δ of H.*

Lemma 7.5. *Assume that X is δ-separated. If one cluster gains or loses in total at least $2kd$ points within a single iteration, then the potential drops by at least δ^2/n during this iteration.*

Proof. If a cluster i gains or loses in total at least $2kd$ points in a single iteration, then there exists another cluster j with which i exchanges at least $2d+1$ points. Let $C_i \subseteq X$ and $C_j \subseteq X$ denote the sets of points assigned to cluster i and cluster j before the point exchange, respectively. Furthermore, let $c_i = \mathrm{cm}(C_i)$ and $c_j = \mathrm{cm}(C_j)$ be the cluster centers before the point exchange. Since X is δ-separated, one of the switching points, say, x, must be at a distance of at least δ from the hyperplane H bisecting c_i and c_j. Assume that x switches from C_i to C_j. Then the potential decreases by at least $\|c_i-x\|^2 - \|c_j-x\|^2 = 2\cdot\|c_i-c_j\|\cdot\mathrm{dist}(x,H) \geq 2\cdot\|c_i-c_j\|\cdot\delta$ by Lemma 7.2.

Now let H' denote the hyperplane bisecting the centers of the clusters i and j in the previous iteration. While H' does not necessarily bisect c_i and c_j, it divides the data points belonging to C_i and C_j correctly. Hence, C_i and C_j lie on different sides of H', which implies that $\|c_i-c_j\| \geq \mathrm{dist}(c_i,H') + \mathrm{dist}(c_j,H')$. Consider the at least $2d+1$ data points switching between clusters i and j. One of them, say, y, must have a

distance of at least δ from H' because X is δ-separated. Let us assume w.l.o.g. that this point y belongs to C_i.

Since y is in C_i, the average distance of the points in C_i to H' is at least δ/n: even if C_i contained n points and all except y had distance zero to H', the sum of all distances is at least δ because of y's contribution. Since $\text{dist}(c_i, H')$ is equal to the average distance of the points from C_i to H', this means that $\|c_i - c_j\| \geq \delta/n$, so the potential decreases by at least $2 \cdot \|c_i - c_j\| \cdot \delta \geq 2\delta^2/n$. □

Lemma 7.6 ([3])**.** *Let $X \subseteq \mathbb{R}^d$ be a set of at least d points and let H denote an arbitrary hyperplane. Then there exists a hyperplane H' passing through d points of X that satisfies*

$$\max_{x \in X} \text{dist}(x, H') \leq 2d \cdot \max_{x \in X} \text{dist}(x, H).$$

Lemma 7.7. *Let X be a set of σ-smooth points with $n = |X|$. The probability that X is not δ-separated is bounded from above by*

$$n^{2d} \left(\frac{4d\delta}{\sigma} \right)^d.$$

Proof. If X is not δ-separated then there exists a hyperplane H and a subset $Y \subseteq X$ of points with $|Y| = 2d$ such that all points in Y have distance at most δ from H (in fact, there exists even such a set $Y \subseteq X$ with $|Y| = 2d + 1$). According to Lemma 7.6 this implies that there exists a hyperplane H' that passes through d points of Y and from which all points in Y have a distance of at most $2d\delta$. Hence, it suffices to bound the probability that such a hyperplane H' exists.

We apply a union bound over all choices for the set $Y \subseteq X$ with $|Y| = 2d$ and the hyperplane H'. There are $\binom{n}{2d}$ choices for the set Y and, once this set is fixed, there are $\binom{2d}{d}$ choices for the points that H' passes through. In total there are at most

$$\binom{n}{2d} \cdot \binom{2d}{d} = \frac{n!}{(n-2d)! \cdot (2d)!} \cdot \frac{(2d)!}{(d!)^2} = \frac{n!}{(n-2d)! \cdot (d!)^2} \leq \frac{n!}{(n-2d)!} \leq n^{2d}$$

choices. Now assume that the set Y and the d points from Y that H' passes trough are fixed. Let $Y_0 \subseteq Y$ denote the d points that H' passes through. We use the principle of deferred decisions and reveal the positions of all points from Y_0. Then also H' is determined (here we use implicitly that any subset of X of size d is linearly independent with probability 1 for σ-smooth sets X). Let $Y_1 = Y \setminus Y_0$. Since H' is fixed and the positions of the points in Y_1 are independent, we obtain

$$\mathbf{Pr}\left[\forall y \in Y_1 : \text{dist}(y, H') \leq 2d\delta\right] = \prod_{y \in Y_1} \mathbf{Pr}\left[\text{dist}(y, H') \leq 2d\delta\right].$$

Each y is a Gaussian random vector with standard deviation σ. Hence, it remains to analyze the probability that such a random vector is within distance δ of some fixed hyperplane H'. Let v denote the normal vector of H'. Then $\text{dist}(y, H') = |y \cdot v|$. One extremely useful property of Gaussian random vectors is that any orthogonal

projection of a Gaussian random vector to an affine subspace is again a Gaussian random vector with the same standard deviation in that subspace. In particular, $y \cdot v$ is a one-dimensional Gaussian random variable with standard deviation σ. Hence, the density of $y \cdot v$ is bounded from above by $1/(\sigma\sqrt{2\pi}) \leq 1/\sigma$. This implies

$$\prod_{y \in Y_1} \mathbf{Pr}\left[\operatorname{dist}(y, H') \leq 2d\delta\right] = \prod_{y \in Y_1} \mathbf{Pr}\left[y \cdot v \in [-2d\delta, 2d\delta]\right] \leq \left(\frac{4d\delta}{\sigma}\right)^{|Y_1|} = \left(\frac{4d\delta}{\sigma}\right)^{d}.$$

Now the union bound over all at most n^{2d} choices for Y_0 and Y_1 concludes the proof. □

7.3 Iterations with Small Cluster Changes

We will now consider sequences of 2^k consecutive iterations in which in every iteration every cluster gains or loses at most $2kd$ points. In the following, we will use the term *configuration of a cluster* to refer to the set of points that are assigned to that cluster.

Lemma 7.8. *Consider a sequence of 2^k consecutive iterations of the k-means method and let $\mathcal{C}_1, \ldots, \mathcal{C}_{2^k+1}$ denote the corresponding sequence of clusterings (including the initial clustering C_1 before the first iteration in the sequence). Then there exists at least one cluster that takes on at least three different configurations in $\mathcal{C}_1, \ldots, \mathcal{C}_{2^k+1}$.*

Proof. During the considered sequence the k-means method encounters $2^k + 1$ clusterings. Assume that every cluster takes on at most two different configurations. Then one clustering must repeat in the sequence $\mathcal{C}_1, \ldots, \mathcal{C}_{2^k+1}$. Since the potential Ψ decreases in each iteration, this cannot happen. □

For two sets A and B let $A \triangle B$ denote their *symmetric difference*, i.e.,

$$A \triangle B := (A \setminus B) \cup (B \setminus A).$$

Definition 7.9. *Let $X \subseteq \mathbb{R}^d$ be a set of points. We say that X is* δ-sparse *if there do not exist pairwise distinct sets $S_1, S_2, S_3 \subseteq X$ with $|S_1 \triangle S_2| \leq 2kd$ and $|S_2 \triangle S_3| \leq 2kd$ for which $\|\operatorname{cm}(S_1) - \operatorname{cm}(S_2)\| \leq \delta$ and $\|\operatorname{cm}(S_2) - \operatorname{cm}(S_3)\| \leq \delta$.*

Lemma 7.10. *Assume that X is δ-sparse. Then every sequence of 2^k consecutive iterations in which in every iteration every cluster gains or loses in total at most $2kd$ points improves the potential by at least δ^2.*

Proof. According to Lemma 7.8, there is one cluster that assumes three different configurations S_1, S_2, and S_3 in this sequence. Due to the assumption in Lemma 7.10, we have $|S_1 \triangle S_2| \leq 2kd$ and $|S_2 \triangle S_3| \leq 2kd$. Hence, due to the definition of δ-sparse, we have $\|\operatorname{cm}(S_i) - \operatorname{cm}(S_{i+1})\| > \delta$ for one $i \in \{1, 2\}$. Combining this with Lemma 7.3 concludes the proof. □

We denote by $\mathcal{B}(z, \varepsilon)$ the d-dimensional hyperball with center $z \in \mathbb{R}^d$ and radius $\varepsilon \geq 0$.

Lemma 7.11. *Let $z \in \mathbb{R}^d$ and $\varepsilon \geq 0$ be arbitrary and let x be a d-dimensional Gaussian random vector with standard deviation σ and arbitrary mean. Then*

$$\mathbf{Pr}\left[x \in \mathcal{B}(z, \varepsilon)\right] \leq \left(\frac{\varepsilon}{\sigma}\right)^d.$$

Proof. The probability density function of x is bounded from above by $1/(\sqrt{2\pi}\sigma)^d$. The hyperball $\mathcal{B}(z, \varepsilon)$ is contained in a hypercube of side length 2ε and volume $(2\varepsilon)^d$. Hence, by the same reasoning as in Lemma 2.7 it follows that the probability that x lies in $\mathcal{B}(z, \varepsilon)$ is bounded from above by

$$\left(\frac{1}{\sqrt{2\pi}\sigma}\right)^d \cdot (2\varepsilon)^d \leq \left(\frac{\varepsilon}{\sigma}\right)^d. \qquad \square$$

Lemma 7.12. *Let X be a set of σ-smooth points with $n = |X|$. The probability that X is not δ-sparse is bounded from above by*

$$(7n)^{4kd}\left(\frac{4n^3\delta}{\sigma}\right)^d.$$

Proof. Given sets S_1, S_2, and S_3 with $|S_1 \triangle S_2| \leq 2kd$ and $|S_2 \triangle S_3| \leq 2kd$, we can write S_i as the disjoint union of a common ground set $A = S_1 \cap S_2 \cap S_3$ with the set $B_i = S_i \setminus A$. Then $B_1 \cap B_2 \cap B_3 = \emptyset$. Furthermore,

$$B_1 \cup B_2 \cup B_3 = (S_1 \cup S_2 \cup S_3) \setminus A = (S_1 \triangle S_2) \cup (S_2 \triangle S_3).$$

Hence, $|B_1 \cup B_2 \cup B_3| = |(S_1 \triangle S_2) \cup (S_2 \triangle S_3)| \leq 4kd$.

We use a union bound over all choices for the sets B_1, B_2, and B_3. The number of choices for these sets is bounded from above by $7^{4kd}\binom{n}{4kd} \leq (7n)^{4kd}$, which can be seen as follows: We choose $4kd$ candidate points for $B_1 \cup B_2 \cup B_3$, and then for each point, we choose which set(s) it belongs to (it does not belong to all of them, but we allow that it belongs to none of them because otherwise we would not cover the case that $B_1 \cup B_2 \cup B_3$ contains fewer than $4kd$ points). We assume in the following that the sets B_1, B_2, and B_3 are fixed. For $i \in \{1, 2\}$, we can write $\mathrm{cm}(S_i) - \mathrm{cm}(S_{i+1})$ as

$$\left(\frac{|A|}{|A| + |B_i|} - \frac{|A|}{|A| + |B_{i+1}|}\right) \cdot \mathrm{cm}(A) + \frac{|B_i|}{|A| + |B_i|} \cdot \mathrm{cm}(B_i) - \frac{|B_{i+1}|}{|A| + |B_{i+1}|} \cdot \mathrm{cm}(B_{i+1}). \tag{7.1}$$

Let us first consider the case that we have $|B_i| = |B_{i+1}|$ for one $i \in \{1, 2\}$. Then $\mathrm{cm}(S_i) - \mathrm{cm}(S_{i+1})$ simplifies to

$$\frac{|B_i|}{|A| + |B_i|} \cdot (\mathrm{cm}(B_i) - \mathrm{cm}(B_{i+1})) = \frac{1}{|A| + |B_i|} \cdot \left(\sum_{x \in B_i \setminus B_{i+1}} x - \sum_{x \in B_{i+1} \setminus B_i} x\right).$$

Since $B_i \neq B_{i+1}$, there exists a point $y \in B_i \triangle B_{i+1}$. Let us assume without loss of generality that $y \in B_i \setminus B_{i+1}$ and that the positions of all points in $(B_i \cup B_{i+1}) \setminus \{y\}$ are fixed arbitrarily. Then the event that $\|\mathrm{cm}(S_i) - \mathrm{cm}(S_{i+1})\| \leq \delta$ is equivalent to

the event that y lies in a fixed hyperball of radius $(|A| + |B_i|)\delta \leq n\delta$. Hence, the probability is bounded from above by $(n\delta/\sigma)^d \leq (4n^3\delta/\sigma)^d$ according to Lemma 7.11.

Now assume that $|B_1| \neq |B_2| \neq |B_3|$. For $i \in \{1, 2\}$, we set

$$r_i = \left(\frac{|A|}{|A| + |B_i|} - \frac{|A|}{|A| + |B_{i+1}|} \right)^{-1} = \frac{(|A| + |B_i|) \cdot (|A| + |B_{i+1}|)}{|A| \cdot (|B_{i+1}| - |B_i|)}$$

and

$$Z_i = \frac{|B_{i+1}|}{|A| + |B_{i+1}|} \cdot \mathrm{cm}(B_{i+1}) - \frac{|B_i|}{|A| + |B_i|} \cdot \mathrm{cm}(B_i) \, .$$

According to Equation (7.1), the event $\|\mathrm{cm}(S_i) - \mathrm{cm}(S_{i+1})\| \leq \delta$ is equivalent to the event that $\mathrm{cm}(A)$ falls into the hyperball with radius $|r_i|\delta$ and center $r_i Z_i$. Hence, the event that both $\|\mathrm{cm}(S_1) - \mathrm{cm}(S_2)\| \leq \delta$ and $\|\mathrm{cm}(S_2) - \mathrm{cm}(S_3)\| \leq \delta$ can only occur if the hyperballs $\mathcal{B}(r_1 Z_1, |r_1|\delta)$ and $\mathcal{B}(r_2 Z_2, |r_2|\delta)$ intersect. This event occurs if and only if the centers $r_1 Z_1$ and $r_2 Z_2$ have a distance of at most $(|r_1| + |r_2|)\delta$ from each other. Hence,

$$\begin{aligned} &\mathbf{Pr}\left[(\|\mathrm{cm}(S_1) - \mathrm{cm}(S_2)\| \leq \delta) \wedge (\|\mathrm{cm}(S_2) - \mathrm{cm}(S_3)\| \leq \delta)\right] \\ &\leq \mathbf{Pr}\left[\|r_1 Z_1 - r_2 Z_2\| \leq (|r_1| + |r_2|)\delta\right] \, . \end{aligned}$$

After some algebraic manipulations, we can write the vector $r_1 Z_1 - r_2 Z_2$ as

$$\begin{aligned} &- \frac{|A| + |B_2|}{|A| \cdot (|B_2| - |B_1|)} \cdot \sum_{x \in B_1} x - \frac{|A| + |B_2|}{|A| \cdot (|B_3| - |B_2|)} \cdot \sum_{x \in B_3} x \\ &+ \left(\frac{|A| + |B_1|}{|A| \cdot (|B_2| - |B_1|)} + \frac{|A| + |B_3|}{|A| \cdot (|B_3| - |B_2|)} \right) \cdot \sum_{x \in B_2} x \, . \end{aligned}$$

Since $B_1 \neq B_3$, there must be a $y \in B_1 \triangle B_3$. We can assume that $y \in B_1 \setminus B_3$. If $y \notin B_2$, we let an adversary choose all positions of the points in $B_1 \cup B_2 \cup B_3 \setminus \{y\}$. Then the event $\|r_1 Z_1 - r_2 Z_2\| \leq (|r_1| + |r_2|)\delta$ is equivalent to y falling into a fixed hyperball of radius

$$\begin{aligned} &\left| \frac{|A| \cdot (|B_2| - |B_1|)}{|A| + |B_2|} \right| \cdot (|r_1| + |r_2|)\delta \\ &= \left| (|B_2| - |B_1|) \cdot \left(\left| \frac{|A| + |B_1|}{|B_2| - |B_1|} \right| + \left| \frac{|A| + |B_3|}{|B_3| - |B_2|} \right| \right) \right| \delta \leq 4kdn\delta \leq 4n^3\delta, \end{aligned}$$

where we used for the second to last inequality that $|B_2| - |B_1| \leq 2kd$ and that $|A| + |B_i| \leq n$ for every $i \in \{1, 2, 3\}$. The probability of this event is thus bounded from above by $(4n^3\delta/\sigma)^d$.

It remains to consider the case that $x \in (B_1 \cap B_2) \setminus B_3$. Also in this case we let an adversary choose the positions of the points in $B_1 \cup B_2 \cup B_3 \setminus \{x\}$. Now the event $\|r_1 Z_1 - r_2 Z_2\| \leq (|r_1| + |r_2|)\delta$ is equivalent to x falling into a fixed hyperball of radius

$$\left| - \frac{|A| + |B_2|}{|A| \cdot (|B_2| - |B_1|)} + \left(\frac{|A| + |B_1|}{|A| \cdot (|B_2| - |B_1|)} + \frac{|A| + |B_3|}{|A| \cdot (|B_3| - |B_2|)} \right) \right|^{-1} \cdot (|r_1| + |r_2|)\delta$$

$$= \left| \frac{|A| \cdot (|B_3| - |B_2|)}{|A| + |B_2|} \right| \cdot (|r_1| + |r_2|)\delta$$
$$= \left| (|B_3| - |B_2|) \cdot \left(\left| \frac{|A| + |B_1|}{|B_2| - |B_1|} \right| + \left| \frac{|A| + |B_3|}{|B_3| - |B_2|} \right| \right) \right| \delta \leq 4kdn\delta \leq 4n^3\delta.$$

Hence, the probability is bounded from above by $(4n^3\delta/\sigma)^d$ also in this case.

This concludes the proof because there are at most $(7n)^{4kd}$ choices for B_1, B_2, and B_3 and, for every choice, the probability that both $\|\mathrm{cm}(S_1) - \mathrm{cm}(S_2)\| \leq \delta$ and $\|\mathrm{cm}(S_2) - \mathrm{cm}(S_3)\| \leq \delta$ is at most $(4n^3\delta/\sigma)^d$. □

7.4 Proof of Theorem 7.1

Lemma 7.13. *Let $D_{\max} := \sigma\sqrt{8kd\ln(n)} + 1$ and let X be a set of n Gaussian random vectors in $\mathbb{R}^d$ with mean values in $[0,1]^n$ and standard deviation σ (i.e., X is a σ-smooth point set). Let $\mathcal{F}$ denote the event that $X \not\subseteq [-D_{\max}, D_{\max}]^d$. Then*

$$\mathbf{Pr}[\mathcal{F}] \leq \frac{1}{n^{3kd}}.$$

We need the following lemma before we can prove Lemma 7.13.

Lemma 7.14. *Let X be a Gaussian random variable with mean 0 and standard deviation 1. Then $\mathbf{Pr}[|X| > x] \leq \exp(-x^2/2)$ for all $x \geq 1$.*

Proof. The density function of X is symmetric around 0. Observe that in the following calculation $t \geq x \geq 1$:

$$\mathbf{Pr}[|X| > x] = 2\int_x^\infty \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t \leq \frac{2}{\sqrt{2\pi}} \int_x^\infty t \cdot \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t$$
$$= \frac{\sqrt{2}}{\sqrt{\pi}} \cdot \Big[-\exp(-t^2/2)\Big]_x^\infty = \frac{\sqrt{2} \cdot \exp(-x^2/2)}{\sqrt{\pi}} \leq \exp(-x^2/2). \qquad \square$$

Proof of Lemma 7.13. We use a union bound over all nd coordinates of the points in X. Let x be such a coordinate. Then x is a Gaussian with mean $\mu \in [0,1]$ and standard deviation σ. We apply Lemma 7.14 to the random variable $y = (x - \mu)/\sigma$, which is a Gaussian with mean 0 and standard deviation 1:

$$\mathbf{Pr}[|x| > D_{\max}] \leq \mathbf{Pr}\left[|x - \mu| > \sigma\sqrt{8kd\ln(n)}\right] = \mathbf{Pr}\left[|y| > \sqrt{8kd\ln(n)}\right]$$
$$\leq \exp(-4kd\ln(n)) = \frac{1}{n^{4kd}} \leq \frac{1}{nd \cdot n^{3kd}}.$$

Now the lemma follows from the union bound over the nd coordinates. □

In the following we will use several inequalities that are only true for sufficiently large values of n (i.e., n must be larger than a certain constant). We will assume without further mention that n is sufficiently large and that $2 \leq d \leq n$.

Let Δ denote the smallest total potential decrease made by any sequence of 2^k consecutive iterations of the k-means method.

Lemma 7.15. *For every $\varepsilon \geq 0$,*

$$\mathbf{Pr}\left[\Delta \leq \varepsilon\right] \leq \frac{n^{16k}\varepsilon}{\sigma^2}.$$

Proof. If X is $\sqrt{\varepsilon}$-sparse and $\sqrt{n\varepsilon}$-separated then according to Lemma 7.10 and Lemma 7.5 every sequence of 2^k iterations decreases the potential by at least ε. Hence,

$$\begin{aligned}
\mathbf{Pr}\left[\Delta \leq \varepsilon\right] &\leq \mathbf{Pr}\Big[X \text{ is not } \sqrt{\varepsilon}\text{-sparse or } X \text{ is not } \sqrt{n\varepsilon}\text{-separated}\Big] \\
&\leq \mathbf{Pr}\Big[X \text{ is not } \sqrt{\varepsilon}\text{-sparse}\Big] + \mathbf{Pr}\Big[X \text{ is not } \sqrt{n\varepsilon}\text{-separated}\Big] \\
&\leq (7n)^{4kd}\left(\frac{4n^3\sqrt{\varepsilon}}{\sigma}\right)^d + n^{2d}\left(\frac{4d\sqrt{n\varepsilon}}{\sigma}\right)^d \\
&\leq 2 \cdot (7n)^{4kd}\left(\frac{4n^3\sqrt{\varepsilon}}{\sigma}\right)^d \\
&= 2 \cdot \left(\frac{(7n)^{4k} \cdot 4n^3\sqrt{\varepsilon}}{\sigma}\right)^d \\
&\leq 2 \cdot \left(\frac{(n^{1/5}n)^{4k} \cdot 4n^3\sqrt{\varepsilon}}{\sigma}\right)^d \\
&\leq \left(\frac{n^{5k+3}\sqrt{\varepsilon}}{\sigma}\right)^d \\
&\leq \left(\frac{n^{8k}\sqrt{\varepsilon}}{\sigma}\right)^d.
\end{aligned}$$

From this we can conclude

$$\mathbf{Pr}\left[\Delta \leq \varepsilon\right] \leq \frac{n^{16k}\varepsilon}{\sigma^2}.$$

For $\frac{n^{8k}\sqrt{\varepsilon}}{\sigma} \leq 1$ this follows from $d \geq 2$ and for $\frac{n^{8k}\sqrt{\varepsilon}}{\sigma} > 1$ it follows because $\mathbf{Pr}\left[\Delta \leq \varepsilon\right]$ is always bounded from above by 1. □

Proof of Theorem 7.1. Without loss of generality we assume that the initial centers are in the convex hull of X (if this is not the case initially then it is true after the first iteration). Remember our assumption that $\sigma \leq 1$. For sufficiently large n, we have

$$\begin{aligned}
D_{\max} &= \sigma\sqrt{8kd\ln(n)} + 1 \\
&\leq \sqrt{8kd\ln(n)} + 1 \\
&\leq 2\sqrt{8n^3} \\
&\leq n^2/2.
\end{aligned}$$

If the failure event $\mathcal{F}$ does not occur (i.e., if $X \subseteq [-D_{\max}, D_{\max}]^d$) and if all centers are in the convex hull of X then the distance between any point from X and any center is at most $2\sqrt{d}D_{\max}$. Hence, the potential is bounded from above by

$$4ndD_{\max}^2 \leq 4nd(n^2/2)^2 \leq n^6.$$

According to the definition of Δ, every sequence of 2^k consecutive iterations decreases the potential by at least Δ. Hence, the number of iterations T can only exceed $2^k t$ if $\Delta \leq n^6/t$ or if $\mathcal{F}$ occurs. According to Lemma 7.13 and Lemma 7.15,

$$\begin{aligned}\mathbf{Pr}\left[T \geq 2^k t\right] &\leq \mathbf{Pr}\left[\mathcal{F}\right] + \mathbf{Pr}\left[\Delta \leq \frac{n^6}{t}\right] \\ &\leq \frac{1}{n^{3kd}} + \frac{n^{16k}n^6}{t\sigma^2} \leq \frac{1}{n^{3kd}} + \frac{n^{22k}}{t\sigma^2}.\end{aligned}$$

Using the worst-case upper bound of $(k^2 n)^{kd} \leq n^{3kd}$ for the number of iterations, this implies the following bound on the expected value of T:

$$\begin{aligned}\mathbf{E}\left[T\right] &= \sum_{t=1}^{n^{3kd}} \mathbf{Pr}\left[T \geq t\right] \\ &\leq 2^k + \sum_{t=1}^{n^{3kd}} 2^k \cdot \mathbf{Pr}\left[T \geq 2^k t\right] \\ &\leq 2^k + \sum_{t=1}^{n^{3kd}} 2^k \left(\frac{1}{n^{3kd}} + \frac{n^{22k}}{t\sigma^2}\right) \\ &\leq 2^{k+1} + \sum_{t=1}^{n^{3kd}} \frac{n^{23k}}{t\sigma^2} \\ &= 2^{k+1} + \frac{n^{23k}}{\sigma^2} + \sum_{t=2}^{n^{3kd}} \frac{n^{23k}}{t\sigma^2} \\ &\leq 2^{k+1} + \frac{n^{23k}}{\sigma^2} + \int_1^{n^{3kd}} \frac{n^{23k}}{t\sigma^2}\, dt \\ &\leq 2^{k+1} + \frac{n^{23k}}{\sigma^2} + \left[\frac{n^{23k}\ln(t)}{\sigma^2}\right]_1^{n^{3kd}} \\ &= O\left(\frac{n^{26k}}{\sigma^2}\right).\end{aligned}$$

This concludes the proof of the theorem. □

In the previous analysis, we did not optimize the degree of the polynomial. The reader has probably noticed that many estimates were rather generous and that a more careful analysis would lead to a polynomial with smaller degree. However, our goal here was only to illustrate the method and the main proof ideas that lead to a bound that is polynomial in n^k and $1/\sigma$.

Bibliography

[1] Sanjeev Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *Journal of the ACM*, 45(5):753–782, 1998.

[2] David Arthur, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the k-means method. *Journal of the ACM*, 58(5), 2011.

[3] David Arthur and Sergei Vassilvitskii. Worst-case and smoothed analysis of the ICP algorithm, with an application to the k-means method. *SIAM Journal on Computing*, 39(2):766–782, 2009.

[4] René Beier, Heiko Röglin, and Berthold Vöcking. The smoothed number of Pareto optimal solutions in bicriteria integer optimization. In *Proceedings of the 12th Intl. Conf. on Integer Programming and Combinatorial Optimization (IPCO)*, pages 53–67, 2007.

[5] René Beier and Berthold Vöcking. Probabilistic analysis of knapsack core algorithms. In *Proceedings of the 15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 468–477, 2004.

[6] René Beier and Berthold Vöcking. Random knapsack in expected polynomial time. *Journal of Computer and System Sciences*, 69(3):306–329, 2004.

[7] René Beier and Berthold Vöcking. An experimental study of random knapsack problems. *Algorithmica*, 45(1):121–136, 2006.

[8] René Beier and Berthold Vöcking. Typical properties of winners and losers in discrete optimization. *SIAM J. Comput.*, 35(4):855–881, 2006.

[9] Tobias Brunsch, Cornelissen, Bodo Manthey, and Heiko Röglin. Smoothed analysis of the successive shortest path algorithm. In *Proceedings of the 24th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1180–1189, 2013.

[10] Tobias Brunsch and Heiko Röglin. Improved smoothed analysis of multiobjective optimization. *Journal of the ACM*, 62(1):4:1–4:58, 2015.

[11] Barun Chandra, Howard J. Karloff, and Craig A. Tovey. New results on the old k-Opt algorithm for the traveling salesman problem. *SIAM Journal on Computing*, 28(6):1998–2029, 1999.

[12] Matthias Englert, Heiko Röglin, and Berthold Vöcking. Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. In *Proceedings of the 18th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1295–1304, 2007.

[13] Mary Inaba, Naoki Katoh, and Hiroshi Imai. Variance-based k-clustering algorithms by Voronoi diagrams and randomization. *IEICE Trans. Information and Systems*, E83-D(6):1199–1206, 2000.

[14] David S. Johnson and Lyle A. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*. John Wiley and Sons, 1997.

[15] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Computational Geometry: Theory and Applications*, 28(2-3):89–112, 2004.

[16] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–104. Plenum Press, New York, 1972.

[17] Walter Kern. A probabilistic analysis of the switching algorithm for the Euclidean TSP. *Mathematical Programming*, 44(2):213–219, 1989.

[18] Bernhard Korte and Jens Vygen. Combinatorial optimization: Theory and algorithms, 2007. 4th edition.

[19] Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015.

[20] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[21] George S. Lueker. Unpublished manuscript, 1975. Princeton University.

[22] Bodo Manthey and Heiko Röglin. Improved smoothed analysis of k-means clustering. In *Proceedings of the 20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 461–470. SIAM, 2009.

[23] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[24] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. 2005.

[25] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. 1995.

[26] George L. Nemhauser and Zev Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15:494–505, 1969.

[27] Christos H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[28] Gerhard Reinelt. TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[29] Heiko Röglin and Shang-Hua Teng. Smoothed analysis of multiobjective optimization. In *Proceedings of the 50th Ann. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 681–690. IEEE, 2009.

[30] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[31] Jan van Leeuwen and Anneke A. Schoon. Untangling a traveling salesman tour in the plane. In *Proceedings of the 7th Intl. Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, pages 87–98, 1981.

[32] Andrea Vattani. k-means requires exponentially many iterations even in the plane. *Discrete and Computational Geometry*, 45(4):596–616, 2011.

[33] Norman Zadeh. A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming*, 5(1):255–266, 1973.