Chapter 0

Introduction

Suppose this was the abstract of a journal paper rather than the introduction to a dissertation. Then it would probably end with some cryptic AMS subject classifications and a few key words. One of them surely would be *randomized complexity of linear programming.* Further suppose that you happen to stumble on this keyword while searching a database for references that have to do with your own work. Assuming that you are a random person, chances are 75% that the paper is not of interest to you.

The reason is that both the terms *randomization* and *complexity* in connection with linear programming (LP) independently allow for two rather different interpretations.

0.1 Complexity

Two major questions have dominated the theory of linear programming for a long time.

(i) Are linear programs solvable in polynomial time? In other words, does linear programming belong to the complexity class **P**?

(ii) How many steps does the simplex method take to solve a linear program?

While the first question has been answered in the affirmative, only partial answers have been found to the second one.

The Turing machine model. It was in 1980 when the russian mathematician Khachyian developed his polynomial *ellipsoid method* for linear programming, thereby showing that LP – formulated as the problem of deciding feasibility – is in **P** [30].[1] Although the ellipsoid method is only an approximation algorithm that converges to a feasible solution, if there is one, Khachyian was able to prove that a polynomial number of iterations (polynomial in the bitlengths of the inputs) suffices to find a solution or certify that none exists. Thus, he constructed a polynomial-time *Turing machine* to solve the problem, and under this measure of complexity, the issue was settled from the point of view of classical complexity theory.

[1]See [25] for an introduction into the ellipsoid method.

The real RAM model. Parallel to this development, people were investigating the complexity of the simplex method. This method has been introduced by Dantzig in the early fifties [14], and it still presents the most efficient method to solve LPs in practice. Unlike the ellipsoid method, simplex is a discrete method. It traces a sequence of vertices on the boundary of the feasible region, and its actions are determined by the combinatorial structure of the LP but not by the bitlengths of its actual coordinates. Therefore, the simplex method never takes more steps than the number of vertices, even if the coordinates have arbitrary encoding lengths, and in the unit-cost model or *random access machine* (RAM) model, this is exactly what one assumes. Motivated by the fact that on a 'real' computer numbers are stored and manipulated within a constant amount of space and time, the real RAM model allows arithmetic operations on any real numbers to be performed in constant time per operation. Only this convenient model allows statements like 'A system of n equations in n variables can be solved in time $O(n^3)$ by Gaussian elimination.' To disable 'dirty tricks' of speeding up computations by encoding huge amounts of information into a single number, the RAM model is usually enhanced with the 'fairness assumption' that unit cost applies only to numbers that are not much larger than the input numbers (what 'not much larger' means, depends on how fair you want to be, of course).

To summarize, in the RAM model, the ellipsoid method is not even a finite algorithm, while one gets *combinatorial* upper bounds on the runtime of the simplex method, and exactly such combinatorial bounds (i.e bounds in the RAM model) will be considered in this thesis. Unfortunately, these bounds on the simplex method are not known to be polynomial, and the same is true for any other combinatorial bounds that exist for LP. We say that LP is not known to be *strongly polynomial.*

0.2 Randomization

The situation concerning the analysis of the simplex algorithm is even worse than suggested above. First of all, talking about 'the' simplex method does not really make sense because it becomes an actual algorithm only via a *pivot rule*, and under many pivot rules (among them the one originally proposed by Dantzig), the simplex method needs an exponential number of steps in the worst case. This was first shown by Klee and Minty [31], thereby destroying any hope that the simplex method might turn out to be polynomial in the end, at least under Dantzig's pivot rule. Later this negative result was extended to many other commonly used pivot rules. Two remedies are apparent and this is where the randomization comes in.

(i) Analyze the *average* performance of the simplex method, i.e. its expected behavior on problems chosen according to some natural probability distribution. A good bound in this model would explain the efficiency of the method in practice.

(ii) Analyze *randomized* methods, i.e. methods which base their decisions on internal coin flips. All the exponential worst case examples rely on the fact that a malicious adversary knows the strategy of the algorithm in advance and therefore can come up with just the input for which the strategy is bad. Randomized strategies cannot be fooled in this easy way, if the measure of complexity is the *maximum expected* number of steps, expectation over the internal coin flips performed by the algorithm.

Average performance. In his pioneering work, Borgwardt has shown that the average behavior of the simplex method is indeed polynomial, for a certain variant called the *shadow vertex* algorithm, and under a certain probability distribution imposed on the input [9]. This breakthrough was followed by a sequence of other results improving on the polynomials, weakening the probabilistic model, or considering other pivot rules. It seems that nowadays the average performance of the simplex method is fairly well understood.

Randomized performance. In suggesting remedy (ii) above (which – as you might guess by now – is the one we treat in this thesis), we have not explicitly mentioned the simplex method but randomized methods in general. This is no accident. In fact, randomized algorithms for solving LP in the RAM model have been proposed that are not simplex, although they have 'converged' to the simplex method over the years. For this, the RAM model needs to be enhanced with the assumption that a random number from the set $\{1, \ldots, k\}$ can be obtained in constant time, for any integer k, where 'random' means that each element is chosen with the same probability $1/k$.

Interestingly, this development started with a deterministic algorithm. Megiddo showed that LPs with fixed number of variables (i.e. fixed dimension) can be solved in time linear in the number n of constraints, which is optimal[36]. However, the dependence on the dimension d was doubly exponential. This was improved in a series of papers, by Dyer [15], Dyer and Frieze [16], Clarkson [11], [12] and Seidel [43], the latter establishing a very simple randomized algorithm with expected runtime $O(d!n)$. (We come back to the remarkable algorithm by Clarkson [12] in the Discussion concluding Chapter 5). It was not noticed that Seidel's algorithm was already close in spirit to a dual RANDOM-FACET-SIMPLEX algorithm, see Chapter 2, but Seidel deserves credit for the subsequent developments, including the results of this thesis.

Then two things happened independently. First, Sharir and Welzl enhanced Seidel's incremental algorithm with a method of reusing in each step much of the information obtained in previous steps. This led to a more efficient algorithm, with runtime $O(d^2 2^d n)$. Moreover, this algorithm was formulated in the abstract framework of *LP-type problems* that made it possible to solve problems more general than linear programming [45].

At the same time, Kalai really applied randomization to the simplex method, and only his result lets remedy (ii) actually appear as a remedy. Kalai was able to prove that the RANDOM-FACET-SIMPLEX algorithm takes an expected *subexponential* number of steps on any linear program [27]. In other words, this variant defeats all the worst case counterexamples that have been developed for the deterministic variants. (Still, it is not a polynomial variant, of course.) Only at this stage it explicitly became clear that Sharir and Welzl's algorithm was actually a dual RANDOM-FACET-SIMPLEX algorithm in disguise, their information reusage scheme corresponding to a dual pivot step. (How explicit the correspondence really is has only recently been shown by Goldwasser who gives a nice survey about the developments we address here [23].)

Motivated by Kalai's result, Sharir and Welzl, together with Matoušek, subsequently proved a subexponential bound on the expected number of iterations their algorithm needs, under a certain condition even in the abstract setting of LP-type problems. However, LP was the only concrete problem that could actually be solved within these time bounds [33].

0.3 Overview and Statement of Results

The main new results of this thesis are the following.

- **Chapter 4, Convex Optimization.** We introduce a class of convex programming problems (including the *polytope distance* problem and the *minimum spanning ball* problem) and develop a concrete *simplex-type* method for solving all the problems in this class. In the framework of LP-type problems, this leads to a unified, linear time algorithm for all problems in this class, when the dimension is fixed. Previously, only specific solutions to some problems were known, like Wolfe's simplex algorithm for quadratic programming [41, section 7.3].

- **Chapter 5, Upper bounds for Abstract Optimization**. We derive *randomized, subexponential* upper bounds for the polytope distance and the minimum spanning ball problem (and more general, for any quadratic programming problem in our class of convex programs). This follows from a subexponential algorithm we develop for *abstract optimization problems* which is an abstract class of problems still substantially more general than LP-type problems. We thus establish subexponential bounds both for concrete geometric problems as well as for general abstract problems (in particular, for all LP-type problems), where prior to this they were only known to apply to LP and some LP-type problems.

- **Chapter 6, Lower Bounds for Abstract Optimization.** We prove a *deterministic, exponential* lower bound for abstract optimization problems, showing that randomization is a crucial tool in solving them.

  We derive a nearly tight lower bound on the performance of the Random-Edge-Simplex algorithm on specific LPs, thereby answering a long-standing open question about the behavior of this algorithm. The result naturally extends to the framework of LP-type problems and the abstract version of Random-Edge-Simplex in this framework.

All the material in this thesis is consequently motivated and derived from the simplex method that we carefully review in Chapter 1.

Turning history upside down, we obtain Sharir and Welzl's algorithm for LP-type problems as an abstraction of the Random-Facet-Simplex algorithm. This means, we pursue an inverse approach in two respects. First, we proceed 'backwards in time' and second, we derive the runtime bounds of [33] for LP by applying the *primal* Random-Facet-Simplex algorithm to the *dual* LP. In this approach, the primitive operations – whose nature and realization remain somewhat obscure in [33] – are just well-understood pivot steps in the simplex method. Chapter 2 introduces algorithms Random-Facet-Simplex and Random-Edge-Simplex, and prepares the ground for the subsequent 'lifting' of these algorithms to the abstract level.

In Chapter 3, we review the concept of LP-type problems and derive the $O(d^2 2^d n)$ bound for LP in the abstract framework. We actually start with a different concept, namely *LP-type systems* which we prove to be essentially equivalent to LP-type problems. LP-type systems naturally arise in our simplex-type approach that builds on the *bases* of an LP, while LP-type problems rather come up in the original – dual – approach that builds on *subproblems*. In fact, *subproblem solvability* is a crucial feature of both. We conclude the chapter by proving that the framework of *abstract polytopes* with objective function as introduced by Adler and Saigal [3] to abstract from properties of linear programming, is closely related to LP-type systems, although less general. However, the spirit of both frameworks turns out to be the same, it boils down to a feature called *locality* in [33].

Chapter 4 presents concrete non-linear LP-type problems and proves that they allow a fast 'pivoting routine' similar to the one performed by the simplex algorithm on LP. Therefore, the resulting solution method is called 'simplex-type'. After having developed such pivoting routines, the problems under consideration are solvable in linear time (in fixed dimension) by the algorithm for LP-type problems that we have derived in Chapter 3. Among the problems that the approach can handle are the polytope distance problem and the minimum spanning ball problem, where we need to prove the latter to have a reformulation as a problem of minimizing a convex function subject to linear constraints; at first sight, such a reformulation is not apparent from the problem.

In Chapter 5 we prove the subexponential bounds for abstract optimization problems (AOP). These problems are obtained by further relaxing the defining properties of LP-type problems. In particular, we will drop the locality condition addressed above and only keep subproblem solvability. As it turns out, the latter is the crucial requirement for the subexponential results. Locality becomes obsolete under the *improvement oracle* that we consider for AOP. This oracle is an abstraction of the pivot step in case of LP resp. the primitive operations in case of LP-type problems. The results on AOPs apply to the geometric problems of Chapter 4 and lead to subexponential bounds there.

Chapter 6 slowly takes us back, all the way to linear programming. We start by showing that randomization provably helps in solving AOPs. More specifically, we prove that no deterministic algorithm can beat a trivial runtime bound, while exactly such a bound has been developed for a randomized algorithm in Chapter 5.

We proceed by reviewing a result of Matoušek, proving that the kind of subexponential analysis performed in Chapter 5 is essentially best possible for the algorithms under consideration, already for LP-type problems [34]. In other words, to obtain better bounds, the algorithm would have to be altered.

Finally, we analyze the behavior of the abstract RANDOM-EDGE-SIMPLEX algorithm on Matoušek's examples. We find that a quadratic upper and nearly quadratic lower bound holds. (This does not mean that RANDOM-EDGE-SIMPLEX is polynomial or even better than RANDOM-FACET-SIMPLEX on other problems – Matoušek's example are of special nature). As the most interesting application of our methods we obtain a lower bound for the actual RANDOM-EDGE-SIMPLEX algorithm on an actual LP, the so-called *Klee-Minty cube*. This is exactly the LP that serves as the basic building block for all known worst-case examples proving exponential behavior of deterministic simplex variants. Therefore, this LP is a natural test problem for randomized variants; the particular question how RANDOM-EDGE-SIMPLEX behaves on it was open since a wrong answer had been suggested in 1982.

Chapter 7 is an appendix containing facts and bounds referenced elsewhere in the text. This is meant to keep the thesis mostly self-contained. If you find some piece of terminology in the text that you don't understand, it might appear in the Appendix.

One final remark: this thesis does not contain any 'big-Oh's, unless they occur in bounds that are standard or that we do not prove (like $O(n^3)$ denoting the time to solve a system of linear equations by Gaussian elimination). All the bounds that we derive contain explicit constants. Just to stress their order of magnitude, 'big-Oh' versions are occasionally added. Only in some places does this rigor require substantial technical effort. Most of the time, the explicit constants are just a consequence of a clean proof.