THREE PUZZLES ON MATHEMATICS, COMPUTATION, AND GAMES

GIL KALAI
HEBREW UNIVERSITY OF JERUSALEM AND YALE UNIVERSITY

ABSTRACT. In this lecture I will talk about three mathematical puzzles involving mathematics and computation that have preoccupied me over the years. The first puzzle is to understand the amazing success of the simplex algorithm for linear programming.

1. INTRODUCTION

The theory of computing and computer science as a whole are precious resources for mathematicians. They bring new questions, new profound ideas, and new perspectives on classical mathematical objects, and serve as new areas for applications of mathematics and of mathematical reasoning. In my lecture I will talk about three mathematical puzzles involving mathematics and computation (and, at times, other fields) that have preoccupied me over the years. The connection between mathematics and computing is especially strong in my field of combinatorics, and I believe that being able to personally experience the scientific developments described here over the last three decades may give my description some added value. For all three puzzles I will try to describe with some detail both the large picture at hand, and zoom in on topics related to my own work.

Puzzle 1: What can explain the success of the simplex algorithm? Linear programming is the problem of maximizing a linear function ϕ subject to a system of linear inequalities. The set of solutions for the linear inequalities is a convex polytope P (which can be unbounded). The simplex algorithm was developed by George Dantzig. Geometrically it can be described by moving from one vertex to a neighboring vertex so as to improve the value of the objective function.

The simplex algorithm is one of the most successful mathematical algorithms. The explanation of this success is an applied, vaguely stated problem, which is connected with computers. The problem has strong relations to the study of convex polytopes, which fascinated mathematicians from ancient times and which served as a starting point for my own research. In Section 2 we will discuss the mathematics of the simplex algorithm, convex polytopes, and related mathematical objects.

Two important elements in studying the mathematics of linear programming are abstractions and reductions. Various abstract forms of linear programming, convex polytopes, and other objects play an important role, and there are very clever and important reductions from one set of problems and objects to others.

If I were required to choose the single most important mathematical explanation for the success of the simplex algorithm, my choice would point to a theorem about another algorithm. I would choose Khachiyan's 1979 theorem asserting that there is a polynomial-time algorithm for linear programming. (Or briefly $LP \in \mathbf{P}$.) Khachiyan's theorem refers to the ellipsoid method, and the answer is

Work supported in part by ERC advanced grant 320924, BSF grant 2006066, and NSF grant DMS-1300120.

given in the language of computational complexity, a language that did not exist when the question was originally raised.

We will concentrate on the study of diameter of graphs of polytopes and the discovery of randomized subexponential variants of the simplex algorithm, I'll mention recent advances: the disproof of the Hirsch conjecture by Santos and the connection between linear programming and stochastic games leading to subexponential lower bounds, discovered by Friedmann, Hansen, and Zwick, for certain pivot rules.

2. Linear programming, polytopes, and the simplex algorithm

2.1. **Linear programming and the simplex algorithm.** A linear programming problem is the problem of finding the maximum of a linear functional (called "a linear objective function") ϕ on d variables subject to a system of n inequalities.

Maximize

$c_1x_1 + c_2x_2 + \cdots c_dx_d$

subject to

$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1d} \leq b_1$

$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2d} \leq b_2$

...

$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nd} \leq b_n$

This can be written briefly as

Maximize c^tx subject to $Ax \leq b$,

where by convention (only here and under protest) vectors are column vectors, $x = (x_1, x_2, \ldots, x_n)$, $b = (b_1, b_2, \ldots, b_n)$, $c = (c_1, c_2, \ldots, c_n)^t$ and $A = (a_{ij})_{1\leq i\leq n, 1\leq j\leq n}$.

The set of solutions to the inequalities is called the *feasible polyhedron* and the simplex algorithm consists of reaching the optimum by moving from one vertex to a neighboring vertex. The precise rule for this move is called "the pivot rule". What we just described is sometimes called the second phase of the algorithm, and there is a first phase where some point of the feasible polyhedron is reached.

Here is an example where $n = 2d$:

Maximize

$x_1 + x_2 + \cdots + x_d$

subject to:

$0 \leq x_i \leq 1, i = 1, 2, \ldots, d$

In this case the feasible polyhedra is the d-dimensional cube. Although the number of vertices is exponential, 2^d, for every pivot rule it will take at most d steps to reach the optimal vertex $(1, 1, \ldots, 1)$.

The study of linear programming and its major applications in economics was pioneered by Kantorovich and Koopmans in the early 1940s. In the late 1940s George Dantzig realized the importance of linear programming for planning problems, and introduced the simplex algorithm for solving linear programming problems. Linear programming and the simplex algorithm are among the most celebrated applications of mathematics. The question can be traced back to a 1827 paper by Fourier.

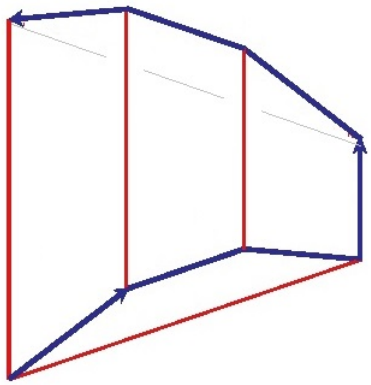

FIGURE 1. The Klee-Minty cube.

2.1.1. *Local to global principle.* We describe now two basic properties of linear programming.

- If ϕ is bounded from above on P then the maximum of ϕ on P is attained at a face of P, in particular there is a vertex v for which the maximum is attained. If ϕ is not bounded from above on P then there is an edge of P on which ϕ is not bounded from above.
- A sufficient condition for v to be a vertex of P on which ϕ is maximal is that v is a *local maximum*, namely $\phi(v) \geq \phi(w)$ for every vertex w which is a neighbor of v.

An *abstract objective function* (AOF) on a polytope P is an ordering of the vertices of P such that every face F of P has a unique local maximum.

Linear programming duality. A very important aspect of linear programming is duality. Linear programming duality associates to an LP problem (given as a maximization problem) with d variables and n inequalities, a dual LP problem (given as a minimization problem) with $n - d$ variables and n inequalities with the same solution. Given an LP problem, the simplex algorithm for the dual problem can be seen as a path-following process on vertices of the hyperplane arrangement described by the entire hyperplane arrangement described by the n inequalities. It moves from one dual-feasible vertex to another, where dual-feasible vertex is the optimal vertex to a subset of the inequalities.

2.2. **Overview.** *Early empirical experience and expectations.* The performance of the simplex algorithm is extremely good in practice. In the early days of linear programming it was believed that the common pivot rules reach the optimum in a number of steps which is polynomial or perhaps even close to linear in d and n. A related conjecture by Hirsch asserted that for polyhedra defined by n inequalities in d variables, there is always a path of length at most $n - d$ between every two vertices. We overview some developments regarding linear programming and the simplex algorithms where by "explanations" we refer to theoretical results that can be seen as giving theoretical support for the excellent behavior of the simplex algorithm, while by "concerns" we refer to results in the opposite direction.

(1) *The Klee-Minty example and worst case behavior (concern 1).* Klee and Minty (1972) found that one of the most common variants of the simplex algorithm is exponential in the worst case. In fact, the number of steps was quite close to the total number of vertices of the feasible

polyhedron (which is combinatorially equivalent to a cube). Similar results for other pivot rules were subsequently found by several authors.

(2) *Klee-Walkup counterexample for the Hirsch Conjecture (concern 2).* Klee and Walkup (1967) found an example of an unbounded polyhedron for which the Hirsch Conjecture fails. They additionally showed that also in the bounded case one cannot realize the Hirsch bound by improving paths. The Hirsch conjecture for polytopes remained open. On the positive side, Barnette and Larman gave an upper bound for the diameter of graphs of d-polytopes with n facets which are exponential in d but linear in n.

(3) $LP \in \mathbf{P}$*, via the ellipsoid method (explanation 1).* In 1979 Khachiyan proved that $LP \in P$ namely that there is a polynomial time algorithm for linear programming. This was a major open problem ever since the complexity classes P and NP where described in the early 1970s. Khachiyan's proof was based on Yudin, Nemirovski and Shor's ellipsoid method, which is not practical.

(4) *Amazing consequences.* Grötschel, Lovász, and Schrijver (1981) found many theoretical applications of the ellipsoid method well beyond its original scope, and found polynomial time algorithms for several classes of combinatorial optimization problems. In particular they showed that semi-definite programming, the problem of maximizing a linear objective function on the set of m by m positive definite matrices, is in **P**.

(5) *Interior points methods (explanation 2).* For a few years there was a feeling that there is a tradeoff between theoretical worst case behavior and practical behavior. This feeling was shattered with Karmarkar's 1984 interior point method and subsequent theoretical and practical discoveries.

(6) *Is there a strongly polynomial algorithm for LP? (Concern 3)* All known polynomial-time algorithms for LP require a number of arithmetic operations which is polynomial in d and n and linear in L, the number of bits required to represent the input. Strongly-polynomial algorithms are algorithms where the number of arithmetic operations is polynomial in d and n and does not depend on L, and no strongly polynomial algorithm for LP is known.

(7) *Average case complexity (explanation 3).* Borgwart (1982) and Smale (1983) pioneered the study of average case complexity for linear programming. Borgwart was able to show polynomial average case behavior for a certain model which exhibit rotational symmetry. In 1986, Adler and Megiddo, Adler, Shamir, and Karp, and Todd were able to prove *quadratic* upper bounds for the simplex algorithm for very general random models that exhibit certain sign invariance. All these results are for a certain pivot rule, first introduced by Gass and Saaty, called the shadow boundary rule.

(8) *Linear-on-average upper bound for monotone path in arrangements (explanation 4)* Haimovich and Adler (1983) proved that the average length of the shadow boundary path from the bottom vertex to the top, for the regions in an arbitrary arrangements of n-hyperplanes in $\mathbb{R}^d$ is linear in d. (This was motivated by Smale's early results on average-case complexity and used in later results.)

(9) *Smooth complexity (explanation 5).* Spielman and Teng (2004) showed that for the shadow-boundary pivot rule, the average number of pivot steps required for a random Gaussian perturbation of variance σ of an arbitrary LP problem is polynomial in d, n and σ^{-1}. (The dependence on d is at least d^5.)

(10) *LP algorithms in fixed dimension.* Megiddo (1984) was the first to find for a fixed value of d a linear time algorithm for LP problems with d variables and n variables. Subsequent simple randomized algorithms were found by Clarkson (1985,1995), Seidel (1991) and Sharir and Welzl (1992). Sharir and Welzl also defined a notion of abstract linear programming problems for which their algorithm applies.

(11) *Quasi polynomial bounds for the diameter (explanation 6).* Kalai (1992) and Kalai and Kleitman (1992), proved a quasipolynomial upper bound for the diameter of graphs of d-polytopes with n facets.
(12) *Sub-exponential pivot rules (explanation 7)* Kalai (1992) and Matoušek, Sharir, and Welzl (1992) proved that there are randomized pivot steps which require in expectation a subexponential number of steps $\exp(K\sqrt{\log nd})$. One of those algorithms is the Sharir-Welzl algorithm.
(13) *Subexponential lower bounds for abstract problems (concern 4)* Matoušek (1994) and Matoušek and Szabó (2006) [**?**] found subexponential lower bound for the number of steps required by two basic randomized simplex pivot rules, for abstract linear programs.
(14) *Santos' counterexample for the Hirsch Conjecture (concern 5).* Santos (2012) found a counterexample to the Hirsch conjecture.
(15) *The connection with stochastic games.* Ludwig (1995) showed that the subexponential randomized pivot rule can be applied to the problem posed by Condon of finding the value of certain stochastic games. For these games this is the best known algorithm.
(16) *Subexponential lower bounds for geometric problems (concern 6).* Building on the connection with stochastic games, subexponential lower bounds for genuine LP problems for several randomized pivot rules were discovered by Friedman, Hansen, and Zwick(2011) [**?**].

Most of the developments listed above are in the very theoretical side of the linear programming research and there are also many other important theoretical aspects. Improving the linear algebra aspects of LP algorithms, and tailoring the algorithm to specific structural and sparsity features of optimization tasks are very important and poses important mathematical challenges. Also of great importance are widening the scope of applications, and choosing the right LP modeling to real-life problems. Of course, there is much theoretical and practical work on special families of LP problems.

2.3. **Complexity 1: P, NP, and LP.**

2.3.1. *Complexity of algorithms.* The complexity of an algorithmic task is the number of steps required by a computer program to perform the task. The complexity is given in terms of the input size, and usually refers to the worst case behavior given the input size. An algorithmic task is in **P** (called "polynomial" or "efficient") if there is a computer program that performs the task in number of steps which is bounded above by a polynomial function of the input size. (In contrast, an algorithmic task which requires an exponential number of steps in terms of the input size is referred to as "exponential" or "intractable".)

The notion of a non-deterministic algorithm is one of the most important notions in the theory of computation. One way to look at non-deterministic algorithms is to refer to algorithms where some or all steps of the algorithm are chosen by an almighty oracle.

Decision problems are algorithmic tasks where the output is either "yes" or "no". A decision problem is in **NP** if when the answer is yes, it admits a non-deterministic algorithm with a polynomial number of steps in terms of the input size. In other words, if for every input for which the answer is "yes," there is an efficient proof demonstrating it, namely, a polynomial size proof that a polynomial time algorithm can verify. An algorithmic task A is **NP**-hard if a subroutine for solving A allows solving any problem in **NP** in a polynomial number of steps. An **NP**-complete problem is an **NP**-hard problem in **NP**.

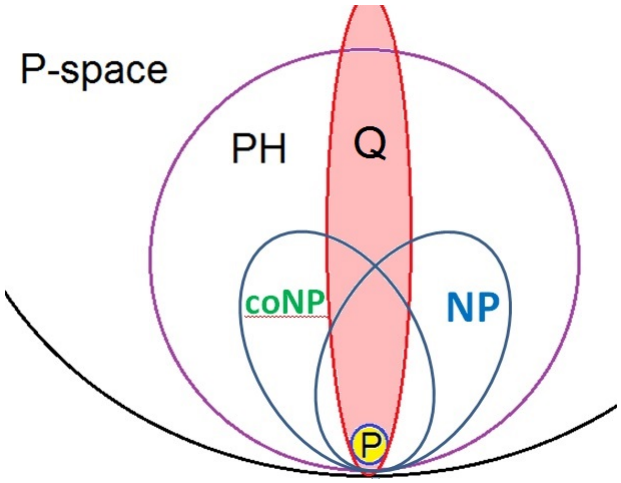

FIGURE 2. The (conjectured) view of some main computational complexity classes. The red ellipse represents efficient quantum algorithms discussed in Section **??**.

Graph algorithms play an important role in computational complexity. PERFECT MATCHING, the agorithmic problem of deciding if a given graph G contains a perfect matching is in **NP** because exhibiting a perfect matching gives an efficient proof that a perfect matching exists. PERFECT MATCHING is in **co-NP** (namely, "not having a perfect matching" is in **NP**) because by a theorem of Tutte, if G does not contain a perfect matching there is a simple efficient way to demonstrate a proof. An algorithm by Edmond shows that PERFECT MATCHING is in **P**. The problem of deciding if G contains a Hamiltonian cycle is also in **NP** - exhibiting a Hamiltonian cycle gives an efficient proof for its existence. However this problem is **NP**-complete.

The papers by Cook (1971), and Levin (1973) introducing the classes **P** and **NP** and raising the conjecture that $\mathbf{P} \neq NP$, and the paper by Karp (1972) identifying 21 central algorithmic problems as **NP**-complete, are among the scientific highlights of the 20th century.

Remark: **P**, **NP**, and **co-NP** are three of the lowest computational complexity classes in the *polynomial hierarchy* **PH**, which is a countable sequence of such classes, and there is a rich theory of complexity classes beyond **PH**. Our understanding of the world of computational complexity depends on a whole array of conjectures: $\mathbf{NP} \neq \mathbf{P}$ is the most famous one. A stronger conjecture asserts that **PH** does not collapse, namely, that there is a strict inclusion between the computational complexity classes defining the polynomial hierarchy.

2.3.2. *The complexity of LP, Khachiyan's theorem, and the quest for strongly polynomial algorithms.* It is known that general LP problems can be reduced to the decision problem to decide if a system of inequalities has a solution. It is therefore easy to see that LP is in **NP**. All we need is to identify a solution. The duality of linear programming implies that LP is in **co-NP** (namely, "not having a solution" is in **NP**).

Theorem 2.1 (Khachiyan (1979))**.** $LP \in \mathbf{P}$. *The ellipsoid method requires a number of arithmetic steps which is polynomial in n, d, and L.*

Here, L is the number of bits required to describe the problem. The dependence on L in Khachiyan's theorem is linear and it was met with some surprise. We note that the efficient demonstration that a system of linear inequalities has a feasible solution requires a number of arithmetic operations which

is polynomial in d and n but do not depend on L. The same applies to an efficient demonstration that a system of linear inequalities is infeasible. Also the simplex algorithm itself requires a number of arithmetic operations that, while not polynomial in d and n in the worst case, does not depend on L. An outstanding open problem is:

Problem 1. Is there an algorithm for LP that requires a number of arithmetic operations which is polynomial in n and d and does not depend on L.

Such an algorithm is called a strongly polynomial algorithm, and this problem is one of Smale's "problems for the 21st century." Strongly polynomial algorithms are known for various LP problems. The Edmond-Karp algorithm (1972) is a strongly polynomial algorithm for the maximal flow problem. Tardos (1986) proved that fixing the feasible polyhedron (and even fixing only the matrix A used to define the inequalities) there is a strongly polynomial algorithm independent of the objective function (and the vector b).

2.4. **Diameter of graphs of polytopes and related objects.**

2.4.1. *Quasi-polynomial monotone paths to the top.*

Theorem 2.2. *Let P be a d-dimensional simple polyhedron, let ϕ be a linear objective function which is not constant on any edge of P, and let v be a vertex of P. Suppose that there are n active facets w.r.t. v. Then there is a monotone path of length $n \cdot \binom{n}{\log d}$ from v to the top.*

Proof: Let $f(d, n)$ denote the maximum value of the minimum length in a monotone path from v to the top. (Here, "the top" refers to either the top vertex or a ray on which ϕ is unbounded.)

Claim: Starting from a vertex v, in $f(d.k)$ steps one can reach either the top or vertices in at least $k + 1$ active facets.

Proof: Let S be a set of $n - k$ active facets. Remove the inequalities defined by these facets to obtain a new feasible polyhedron Q. If v is not a vertex anymore than v belongs to some facet in S. If v is still a vertex there is a monotone path of length at most $f(d, k)$ from v to the top. If one of the edges in the path leaves P then it reaches a vertex belonging to a facet in S. Otherwise it reaches the top. Now if a monotone path from v (in P) of length $f(d, k)$ cannot take us to to the top, it must take us to a vertex in some facet belonging to every set of $d - k$ facets, and therefore to vertices in at least $k + 1$ facets.

Proof of the Theorem: Order the active facets of P according to their top vertex. In $f(d, n/2)$ steps we can reach from v either the top or a vertex in the top $n/2$ active facets. In $f(d - 1, n - 1)$ steps we can reach the top w of that facet. This will leave us with at most $n/2$ active facets w.r.t. w. This gives

$$f(d, n) \leq 2f(d, n/2) + f(d - 1, n - 1), \tag{2.1}$$

which implies the bounds given by the theorem.

Remarks: 1. A monotone path can be regarded as a non-deterministic version of the simplex algorithm where the pivot steps are chosen by an oracle.

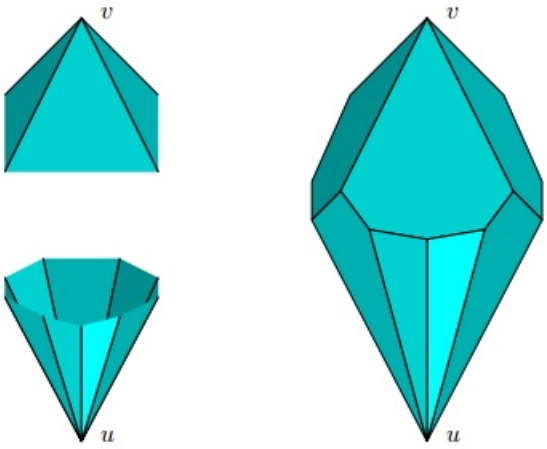

FIGURE 3. A spindle

2.4.2. *Reductions, abstractions and Hahnle's conjecture.* Upper bounds for the diameter are attained at *simple* d-polytopes, namely d-polytopes where every vertex belongs to exactly d facets. A more general question deals with the dual graphs for triangulations of $(d-1)$-spheres with n vertices. All the known upper bounds apply for dual graphs of pure normal $(d-1)$ simplicial complexes. Here "pure" means that all maximal faces have the same dimension and "normal" means that all links of dimension one or more are connected. An even more general framework was proposed by Eisenbrand, Hahnle, Razborov, and Rothvoss.

Problem 2. Consider t pairwise-disjoint nonempty families of degree n monomials with d variables $F_1, F_2, \ldots, F_t$ with the following property: For every $1 \leq i < j < k \leq t$ if $m_i \in F_i$ and $m_k \in F_k$ then there is a monomial $m_j \in F_j$, such that m_j divides the greatest common divisor of m_i and m_k. How large can t be?

A simple argument shows that the maximum denoted by $g(d,n)$ satisfies relation (2.1).

Conjecture 3 (Hahnle (2010) [**?**])**.** $g(d,n) \leq d(n-1)+1$

One example of equality is taking F_k be all monomials $x_{i_1}x_{i_2}\cdots x_{i_d}$ with $i_1+i_2+\cdots+i_d = k-d+1$, $k = d, d+1, \ldots, kd$. Another example of equality is to let F_k be a single monomial of the form $x_i^{d-\ell}x_{i+1}^{\ell}$. ($i = \lfloor k/d \rfloor$ and $\ell = k(\mod d)$.)

2.5. **Santos' counterexample.** The d-step conjecture is a special case of the Hirsch conjecture known to be equivalent to the general case. It asserts that a d-polytope with $2d$ facets has diameter at most d. Santos formulated the following strengthening of the d-step conjecture.

Conjecture 4 (Spindle working Conjecture:)**.** Let P be a d-polytope with two vertices u and v such that every facet of P contains exactly one of them. (Such a polytope is called a d-spindle.) Then the graph-distance between u and v (called simply the length of the spindle) is at most d.

and proved

Theorem 2.3 (Santos (2014)[**?**])**.** (i) *The spindle conjecture is equivalent to the Hirsch conjecture. More precisely, if there is a d-spindle with n facets and length greater than d then there is a counter-example to the Hirsch conjecture of dimension $n-d$ and with $2n-2d$ facets.*

(ii) *(2) There is a 5-spindle of length 6.*

The initial proof of part (2) had 48 facets and 322 vertices, leading to a counterexample in dimension 43 with 86 facets and estimated to have more than a trillion vertices. Matschke, Santos and Weibel (2015) [**?**] found an example with only 25 facets leading to a counterexample of the Hirsch conjecture for a 20-polytope with 40 facets and 36,442 vertices.

An important lesson from Santos' proof is that although reductions are available to simple polytopes and simplicial objects, studying the problem for general polytopes has an advantage. In the linear programming language this tells us that degenerate problems are important. This lesson about reductions is important in other contexts as well.

Problem 5. Find an abstract setting for the diameter problem for polytopes which will include graphs of general polytopes, dual graphs for normal triangulations, and families of monomials.

2.6. **Complexity 2: Randomness in algorithms.** One of the most important developments in the theory of computing is the realization that adding an internal randomness mechanism can enhance the performance of algorithms. Some early appearance to this idea came with Monte Carlo methods by Ulam, von Neumann and Metropolis, and a factoring algorithm by Berlekamp. Since the mid 1970s, and much influenced by Michael Rabin, randomized algorithms have become a central paradigm in computer science.

One of the great achievements were the polynomial time randomized algorithms for testing primality by Solvay-Strassen (1977)and Rabin (1980) Rabin's algorithm was related to an earlier breakthrough – Miller's algorithm for primality (1976)which was polynomial time conditioned on the validity of the generalized Riemann hypothesis. As indicated in Rabin's paper, the newly randomized algorithms were not only theoretically efficient but also practically excellent! Rabin's paper thus gave "probabilistic proofs" that certain large numbers are primes and this was a new kind of mathematical proof. A deterministic polynomial algorithm for primality was achieved by Agrawal, Kayal, and Saxena (2004).

Lovász (1979) (building on earlier works by Tutte and Edmonds) offered a randomized efficient algorithm for perfect matching in bipartite graphs: Associate to a bipartite graph G with n vertices on each side, its generic $n \times n$ adjacency matrix A, where a_{ij} is zero if the ith vertex on one side is not adjacent to the jth vertex on the other side, and a_{ij} is a variable otherwise. Note the determinant of A is zero if and only if G has no perfect matching. This can be verified with high probability by replacing a_{ij} with random mod p elements for a large prime p.

We have ample empirical experience and some theoretical support to the fact that pseudo-random number generators are practically sufficient for randomized algorithm. We have also strong theoretical supports that weak and imperfect sources of randomness are sufficient for randomized algorithms.

A class of randomized algorithms which are closer to early Monte Carlo algorithms and also to randomized algorithms for linear programming, are algorithms based on random walks. Here are two examples: counting the number of perfect matching for a general graph G is a $\#\mathbf{P}$-complete problem. Jerrum and Sinclair (1989) and Jerrum, Sinclair and Vigoda (2001) found an efficient random-walk based algorithm for estimating the number of perfect matching up to a multiplicative constant $1 + \epsilon$. Dyer, Frieze, and Kannan (1991) found an efficient algorithm based on random walks to estimate the volume of a convex body in $\mathbb{R}^d$. Both these algorithm rely in a strong way on the ability to prove a spectral gap (or "expansion") for various Markov chains. Approximate sampling is an important subroutine in the algorithms we have just mentioned and we can regard exact or approximate sampling as an important algorithmic task on its own, as the ability to sample is

theoretically and practically important. We mention algorithms by Aldous-Broder (1989) and Wilson (1996) for sampling spanning trees and an algorithm by Randall and Wilson (1999) for sampling configurations of the Ising models.

Remark: The *probabilistic method* – applying probabilistic methods even for problems with no mention of probability, led to major developments in several other mathematical disciplines. In the area of combinatorics the probabilistic method is especially powerful. (See the book by Alon and Spencer (1992).) Joel Spencer and the author (reluctantly and vaguely) conjecture that for every application of the probabilistic method to prove the existence of a combinatorial object, the object in question can be found by an efficient algorithm. (The algorithm can be a randomized algorithm.)

2.7. **Subexponential randomized simplex algorithms.** We start with the following simple observation.

Observation: Consider the following two sequences. The first sequence is defined by a_1 and $a_{n+1} = a_n + a_{n/2}$, and the second sequence is defined by $b_1 = 1$ and $b_{n+1} = b_n + (b_1 + \cdots b_n)/n$. Then

$$a_n = n^{\theta(\log n)}, \text{ and } b_n = e^{\theta(\sqrt{n})}.$$

Next, we describe two basic randomized algorithms for linear programming.

RANDOM EDGE: Choose an improving edge uniformly at random and repeat.

RANDOM FACET: Given a vertex v, choose a facet F containing v uniformly at random. Use the algorithm recursively inside F reaching its top w, and repeat.

subexponential RANDOM FACET (along with some variants) is the first strongly subexponential algorithm for linear programming, as well as the first subexponential pivot rule for the simplex algorithm.

Theorem 2.4. *Let P be a d-dimensional simple polyhedron, let ϕ be a linear objective function which is not constant on any edge of P, and let v be a vertex of P. Suppose that there are n active facets w.r.t. v. Then random-facet requires at most*

$$\exp(K \cdot \sqrt{n \log d}) \tag{2.2}$$

steps from v to the TOP.

Proof: Write $g(d, n)$ for the expected number of pivot steps. The expected number of pivot step to reach w is bounded above by $g(d-1, n-1)$. With probability i/d, w is the i lowest top of active facets containing v. (The number of those, can be d or $d-1$ for v and $d-1$ later when we reach a facet's top. In the computation we can assume it is $d-1$.) This gives

$$g(d, n) \leq g(d-1, n-1) + \frac{1}{d-1} \sum_{i=1}^{d-1} g(d, n-i).$$

This recurrence relation leads (with some effort) to equation (2.2). □

Note that the argument applies to abstract objective functions on polyhedra. (And even, in greater generality, to abstract LP problems as defined by Sharir-Welzl and to duals of shellable spheres.)

The appearance of $exp\sqrt{n}$ is related to our observation on the sequence b_n. As a matter of fact, for the number of steps $G(d)$ for abstract objective functions in the discrete d-sphere we get the recurrence relation $g(d) = g(d-1) + 1/d(g(1) + g(2) + \cdots + g(d))$.

There are few versions of RANDOM FACET that were analyzed (giving slightly lower or better upper bound). For the best known one see Hansen and Zwick (2015) [**?**]. There are a few ideas for improved

versions: we can talk about a random face rather than a random facet, to randomly walk up and down before setting a new threshold, and to try to learn about the problem and improve the random choices. The powerful results about lower bounds suggest cautious pessimism.

2.7.1. *Lower bounds for abstract problems.* As we will see the hope for better upper bounds for RANDOM FACET and related randomized versions of the simplex algorithm were faced with formidable examples to the contrary.

Theorem 2.5 (Matoušek (1994))**.** *There exists an abstract objective function on the d-cube on which* RANDOM FACET *requires on expectation at least* $\exp(C\sqrt{d})$ *steps.*

Matoušek describes a large class of AOF's and showed his lower bound to hold in expectation for a randomly chosen AOF. Gärtner proved (2002) that for geometric AOF in this family, RANDOM FACET requires an expected quadratic time.

Theorem 2.6 (Matoušek and Szabó (2006) [**?**])**.** *There exists AOF on the d-cube on which* RANDOM EDGE *requires on expectation at least* $\exp(Cd^{1/3})$ *steps.*

2.8. **Games 1: Stochastic games, their complexity, and linear programming.**

2.8.1. *The complexity of Chess and Backgammon.* Is there a polynomial time algorithm for chess? Well, if we consider the complexity of chess in terms of the board size then "generalized chess" is very hard. It is **P-space**-complete. But if we wish to consider the complexity in terms of the number of all possible positions (which for "generalized chess" is exponential in the board size), given an initial position, it is easy to walk on the tree of positions and determine, in linear number of steps, the value of the game. (Real life chess is probably intractable, but we note that Checkers was solved.)

Now, what about backgammon? This question represents one of the most fundamental open problems in algorithmic game theory. The difference between backgammon and chess is the element of luck; in each position your possible moves are determined by a roll of two dice.

Remark: Chess and backgammon are games with complete information and their value is achieved by pure strategies. One of the fundamental insights of game theory is that for zero-sum games with incomplete information, optimal strategies are mixed, namely they are described by random choice between pure strategies. For mixed strategies, von Neumann's 1928 max min theorem asserts that a zero sum game with incomplete information has a value. An optimal strategy for rock-paper-scissors game is to play each strategy with equal probability of 1/3. An optimal strategy for two-players poker (heads on poker) is probably much harder to find.

2.8.2. *Stochastic games and Ludwig's theorem.* A simple stochastic game is a two player zero-sum game with complete information, described as follows: We are given one shared token and a directed graph with two sink vertices labeled '1' and '2' which represent winning positions for the two players respectively. All other vertices have outdegree 2 and are labelled either by a name of a player or as "neutral". In addition, one vertex is the start vertex. Once the token is on a vertex, the player with the vertex labelling moves, and if the vertex is neutral then the move is determined by a toss of a fair coin. Backgammon is roughly a game of this type. (The vertices represent the player whose turn is to play and the outcome of the two dice, and there are neutral vertices representing the rolling of the dice. The outdegrees are larger than two but this does not make a difference.) This class of games was introduced by Condon in 1992. If there is only one player, the game turns into a one-player game with uncertainty, which is called a Markov decision process. For Markov decision processes finding the optimal strategy is a linear programming problem.

Theorem 2.7 (Ludwig (1995))**.** *There is a subexponential algorithm simple stochastic games*

The basic idea of the proof is the following: once the strategy of player 2 is determined the game turns into a Markov decision process and the optimal strategy for player 1 is determined. Player one has an optimization problem over the discrete cube whose vertices represent the choices of player two in each vertex labeled by '2'. The crucial observation is that this optimization problem defines an abstract objective function (with possible equalities) and therefore we we can apply RANDOM FACET.

A more general model of stochastic games with incomplete information was introduced by Shapley in 1953. There at each step the two players choose actions independently from a set of possibilities and their choices determine a reward and a probability distribution for the next state of the game.

Problem 6.
- (Think about backgammon.) Is there a polynomial-time algorithm for finding the value of simple stochastic games.
- Can the problem of finding the sink in a unique sink acyclic orientation of the d-cube be reduced to finding the value of a simple stochastic game?
- Is there a polynomial-time algorithm (or at least, subexponential algorithm) for finding an equilibrium point for a stochastic two-player game (with complete information)?
- Is there a polynomial-time algorithm (or at least, subexponential algorithm) for finding an equilibrium point for a stochastic game with complete information with a fixed number of players?
- (Think about two-player poker) Is there a polynomial-time algorithm (or at least, subexponential) for finding the value of a stochastic zero sum game with incomplete information?

Remark: What is the complexity for finding objects guaranteed by mathematical theorems? Papadimitriou (1994) developed complexity classes and notions of intractability for mathematical methods and tricks! (Finding an efficiently describable object guaranteed by a mathematical theorem cannot be NP-complete (Megiddo (1988).)A motivating conjecture that took many years to prove (in a series of remarkable papers) is that Nash equilibria is hard with respect to PPAD, one of the aforementioned Papadimitriou classes.

Problem 7. How does the problems of finding the sink in a unique sink acyclic orientation of the cube, and solving an abstract LP problem, fit into Papadimitriou's classes?

2.9. **Lower bounds for geometric LP problems via stochastic games.** In this section I discuss the remarkable works of Friedmann, Hansen, and Zwick (2011,2014) [**?**, **?**]. We talked (having the example of backgammon in mind) about two-players stochastic games with complete information. (Uri Zwick prefers to think of those as "games with two and a half players" with nature being a non strategic player rolling the dice.) The work of Friedmann, Hansen, and Zwick starts by building 2-player parity games on which suitable randomized policy-iteration algorithms perform a subexponential number of iterations. Those games are then transformed into 1-player Markov Decision Processes (MDPs) (or $1\frac{1}{2}$-player in Uri's view) which correspond to concrete linear programs. They showed a concrete LP problem, where the feasible polyhedron is combinatorially a cube, on which RANDOM FACET takes an expected number of $e^{\tilde{\Theta}(\sqrt{d})}$ steps. A similar but even more involved argument gives an expected of $\exp(\tilde{\Theta}(d^{1/3})$ steps for RANDOM EDGE . This method can be seen as a new generation of powerful Klee-Minty type cubes, based on games.

Let me note that very recently a quasi-polynomial algorithm for parity games was found. (Even earlier a deterministic subexponential algorithm was known for them.) So far, it does not seem that the algorithm extend to simple stochastic games or has some implications for linear programming.

2.10. **Discussion.** Is our understanding of the success of the simplex algorithm satisfactory? Are there better practical algorithms for semidefinite and convex programming? Is there a polynomial upper bound for the diameter of graphs of d-polytopes with n edges? (Or at least some substantial improvements of known upper bounds). Is there a strongly polynomial algorithm for LP? Perhaps even a strongly polynomial variant of the simplex algorithm? What is the complexity of finding a sink of an acyclic unique-sink orientation of the discrete cube? Are there other new interesting efficient, or practically good, algorithms for linear programming? What is the complexity of stochastic games? Can a theoretical explanation be found for other practically successful algorithms? (Here, SAT solvers for certain classes of SAT problems, and deep learning algorithms come to mind.) Are there good practical algorithms for estimating the number of matchings in graphs? for computing volumes for high dimensional polytopes? We also face the ongoing challenge of using linear programming and optimization as a source for deriving further questions and insights into the study of convex polytopes, arrangements of hyperplanes, geometry and combinatorics.

E-mail address: KALAI@MATH.HUJI.AC.IL