OPTIMA 81

Mathematical Programming Society Newsletter

Steve Wright

MPS Chair's Column

November 20, 2009. Along with so many colleagues, I am deeply saddened by the news of Paul Tseng's disappearance while on a kayak trip on the Jinsha River in China on August 13, 2009. Many of us have many wonderful memories of times spent with Paul – visits, collaborations, conversations, conference social events – and many have been influenced by his scholarly, incisive, and prolific research. A workshop in Paul's honor will take place at Fudan University next May (see announcement in this issue), and an article on Paul will appear in the next issue of *Optima*. A Wikipedia entry about Paul's life and work can be found at http://en.wikipedia.org/wiki/Paul_Tseng.

In MPS news, we have introduced a new section on the web site mathprog.org containing links to faculty job listings in optimization and related areas. Click on the "Jobs" tag at the top of the page, and send any listings you wish to advertise on the site to webmaster@mathprog.org.

The increasing visibility of optimization in other areas of mathematics, science, and engineering has become ever more apparent in recent months. Optimization is being recognized more and more as an important tool for modeling and analysis and as a key to many interdisciplinary research efforts. I illustrate the point with two recent developments. First, the Institute for Pure and Applied Mathematics (IPAM) at UCLA is organizing a three-month program entitled "Modern Trends in Optimization and its Applications," consisting of a series of workshops in Fall 2010. Second, the optimization group at Wisconsin was selected to be one of the five "themes" in the Wisconsin Institutes of Discovery, an interdisciplinary research institute under construction on the UW-Madison campus and funded with $150m of private, state, and alumni research foundation grants. No doubt there are other compelling recent examples of the expanding reach of optimization.

Optimization is naturally an outward-looking discipline. It challenges domain scientists to articulate what it is they need to learn from a model and to evaluate the strengths and weaknesses of various formulations. Conversely, opportunities for fundamental research in optimization grow, as optimizers are challenged to provide new theory and algorithms for the large and complex problems that arise in new collaborations. Interdisciplinary initiatives provide us with exciting opportunities to collaborate with top researchers in other fields. Each success increases the visibility of optimization and makes it easier to find new openings and collaborations in new areas.

Economics provided much of the initial impetus for our discipline. The "zeroth ISMP" took place at the Cowles Commission for Research in Economics in Chicago in 1949; the proceedings can be found at http://cowles.econ.yale.edu/P/cm/m13/. Sixty years later, in the wake of financial and economic crises, the need for rigorous analytical tools in economics and finance is growing once again. Tools based on recent developments in stochastic optimization, robust optimization, complementarity, and algorithmic game theory will be particularly important. The use of optimization in operations research and engineering has grown steadily over the years, in such specific areas as process control, engineering design, and logistics. But it is the past few years that have seen a striking expansion of the application space, into such areas as structural biology and bioinformatics, signal and image processing (including compressed sensing), computational learning, the design and control of electrical grids, and medical applications (e.g. cancer treatment), to name just a few.

The applications that gave rise to our field remain as important as ever, and the vitality and variety of new research topics – both fundamental and interdisciplinary – continues to grow. There has never been a more exciting time to be an optimizer!

Note from the Editors

The topic of Optima 81 is "Counting and Estimating Lattice Points". Instead of the usual format consisting of one scientific contribution and the discussion column, we are very pleased to publish two scientific contributions presenting the subject from different perspectives. The result is a very detailed picture of the topic.

Contents of Issue 81 / December 2009

Jesús A. De Loera

Counting and Estimating Lattice Points: Tools from Algebra, Analysis, Convexity, and Probability

This article gives an overview of the mathematical foundations and the computational complexity of several algorithms for counting lattice points inside polyhedral sets. Given a polytope $P = \{x : Ax = b,\ x \geq 0\}$, specified by a $d \times n$ integral matrix A and an integral d-vector b, our goal is to *count or at least estimate how many lattice* *points are inside P.* In some applications we are in fact interested on the *counting function* $\phi_A(b) = \#\{x : Ax = b, x \geq 0, \ x \text{ integral}\}$, where b can be taken as a vector of parameters. In pure mathematics one is in fact interested in formulas for $\phi_A(b)$ in terms of the variables $b_1, \ldots, b_d$. Areas of application where these algorithmic questions often appear include mathematical programming, probability and statistics, and theoretical computer science. We illustrate with several examples the many uses of computing or evaluating $\phi_A(b)$.

Counting lattice point of polytopes is at the foundation of the theory of mathematical optimization. There is a very close algorithmic relationship between optimization on one hand and counting on the other (this is made explicit in [48, 35, 36]). The number of feasible lattice points in a polyhedron is a measure of progress of a branch-and-bound algorithm for solving integer optimization problems. For integer programs in some applications it is desirable to find all optimal solutions [31]. For example, in network flow problems regarding stability or security one may be interested in knowing the number of maximal integral-valued flows or cuts [7]. This is the same as counting the number of lattice points of a *flow polytope* (see [8] and references therein).

In Statistics counting and sampling lattice points becomes useful to make inferences about the statistical models imposed in tabular data (contingency tables). For example, to decide whether two of the traits of the statistic, say, income and race, are independent of each other, several tests of independence depend on counting the integral tables satisfying some marginal conditions (see [38, 45] and their references). Statisticians have developed ad hoc sampling-based estimation methods and in fact counting is closely related to the problem of sampling a point uniformly at random or with respect to other probability distribution models (see [23, 25, 26, 56] and references therein). Of course for an arbitrary polyhedron this is an unrealistic assumption since it would require intractable computation to even jump start such methods. Counting is also related also to statistical mechanics models such as the Ising and Potts models (see, e.g., [54, 66]).

In computer science, coding theory and cryptography have always been important areas of application for lattices and lattice point counting [64, 83]. An emerging new application of lattice point counting is computer program verification and code optimization. The systematic reasoning about a program's runtime behavior, performance, and execution requires specific knowledge of the number of operations and resource allocation within the program. This is important knowledge for the sake of checking correctness as well as for automatically detecting run-time errors, buffer overflows, null-pointer dereferences or memory leaks. This can be modeled as problem of counting points in parametrized regions (see [28, 47, 67, 78] and references therein). For example, *how often is instruction I of the following computer code executed?*

```
void proc(int N, int M)
{
int i,j;
for (i=2N-M; i<= 4N+M-min(N,M), i++)
  for(j=0; j<N-2*i; j++)
    I;
}
```

Clearly, the number of times we reach instruction I depends parametrically on N, M. In terms of these parameters the set of all possible solutions is given by the number of lattice points inside of a parametrized family of polygons. In our toy example these are described by the conditions $\{(i,j) \in \mathbb{Z}^2 : i \geq 2N - M, \ i \leq 4N + M - \min(N, M), \ j \geq 0, \ j - 2i \leq N - 1\}$.

One can ask *why do we restrict ourselves to polyhedra? What are the most complicated regions for which counting lattice points is possible?* When the sets for which we count lattice points are arbitrary the problem gets harder and harder as the dimension of the set grows: Given (a, b, c) positive integers, deciding whether there is a lattice point in the set $\{x : ax^2 + bx = c, \ x \geq 0\}$ is an NP-complete problem [4]. Deciding whether there is a non-negative integer root for arbitrary polynomials inside $\mathbb{Z}[x_1, \ldots, x_9]$ is *undecidable*, i.e., roughly speaking there is no computer algorithm to solve this problem for all instances [57] already for nine variables! The trouble with undecidability indicates one cannot even create an algorithm for finding a bounding box containing all the possible solutions. How about then if we restrict ourselves to the case of bounded sets? This restriction still contains some interesting surprises. Already Gauss proposed a famous counting problem for circles [49]: Suppose you are given a circle of radius r centered at the origin. Clearly volume is an approximation to the number of lattice points, but how good is the approximation of the number of lattice points $N(r)$ as the area of the circle as r grows? In other words, what is the exact order of magnitude for the error function $N(r) - \pi r^2$? The answer is still not known exactly, although it is supposed to be in between $r^{.5}$ and $r^{.6301369863}$ (see, e.g., [50]).

Counting lattice points in (*four-dimensional*) convex bodies is something that credit card cyber-thieves care about too! The reason is the RSA encryption algorithm relies on the difficulty of factorizing a number of the form pq with p, q large primes. Here is a way to factorize via counting lattice points of 4-dimensional balls: For an integer number $n = pq$ consider the 4-dimensional ball $B(n) = \{x \in R^4 : x_1^2 + x_2^2 + x_3^2 + x_4^2 \leq n\}$. Jacobi (see [5, 51]) proved that the number of ways in which a positive integer can be written as a sum of four squares is eight times the sum of its divisors that are not a multiple of four. So, for example there are 744 ways of writing 100 as sum of four squares because $1, 2, 5, 10, 25, 50$ are the only divisors of 100 not divisible by 4. Thus if we know that $n = pq$, and $|B(n)|$ denotes the number of lattice points in $B(n)$ we have $|B(n)| - |B(n-1)| = 8(1 + p + q + n)$. Therefore a factorization of n could be done fast if we knew how to compute $|B(n)|$ fast. Thus the RSA cryptosystems used in internet transactions could be broken if you knew how to quickly count lattice points inside four dimensional balls.

We continue now with an overview of the theory behind counting and estimation of lattice points inside polytopes. We recommend the books [9, 79, 90] as good references for many concepts about polyhedra and lattices used in the sequel. The literature in this area is quite large and due to lack of space will not be able to mention but a fraction of what is available. We have chosen to delve in more details for two particular algorithms because they both have had somewhat closer relationship to mathematical programming.

1 Exact Counting

Counting lattice points inside convex polyhedra is still difficult: We know that when the dimension is an input variable the problem of detecting a lattice point in polyhedra is NP-hard [46]. Already counting $2 \times n$ contingency tables is $\#P$-hard [43]. Many combinatorial problems can be phrased as counting the lattice points of a polyhedron, thus their hardness translates directly to our problem (e.g., counting perfect matchings of bipartite graphs, also known as the computation of the permanent). In particular one can prove it is NP-hard to estimate the number of lattice points of certain polytopes to any polynomial (in the input length) factor (see, e.g., [58]). But, life goes on, *what can we do if we still have the practical need of counting solutions?*

Branch-and-cut techniques and exhaustive enumeration methods which are common in discrete optimization are also useful for enumerating lattice points. In both approaches one generates a search tree to list all solutions, but branch-and-cut is aided by linear programming relaxations to detect infeasible subproblems and do special pruning (see [31, 3]). One way to attack well-structured problem (such as binary restricted) is to use powerful techniques from constraint programming techniques (see [76] in this issue and references therein). Other search trees techniques have been developed for binary problems (see [18, 24, 3] and references therein). Despite their enormous success, enumeration method can get stuck in surprisingly small problems (e.g., hard knapsack problems such as those in [1]). Enumeration techniques are not useful to derive formulas as they depend on the size of the right-hand-side vector b. During the 1980's and 1990's a new breed of algorithms and counting formulas were created. They rely on deep algebraic and analytic properties of polyhedra.

Some authors have emphasized the algebraic-geometric nature of the lattice points of polyhedra via the theory of toric varieties [82, 39]. Given a polytope $P = \{x : Ax = b,\ x \geq 0\}$, defined by the matrix A with integer entries, the *toric ideal* I_A is the polynomial ideal generated by the binomials of the form $x^u - x^v$ such that $A(u-v) = 0$ (see [82, 85] and references therein). If we find a finite set of binomials, a Gröbner basis is $G_A = \{x^{u_1} - x^{v_1}, x^{u_2} - x^{v_2}, \dots, x^{u_k} - x^{v_k}\}$ that generates I_A (Gröbner bases in general can be computed with the well-known Buchberger algorithm [30]). It is well-known that the vectors $\Gamma = \{u_1 - v_1, u_2 - v_2, \dots, u_k - v_k\}$ will have the following properties: For the polytope $P = \{x : Ax = b,\ x \geq 0\}$, one can form a connected directed graph whose vertices are the lattice points of P with a unique sink. We connect any pair z_1, z_2 of lattice points by an edge if there is a vector $u - v$ in Γ such that $z_1 - u + v = z_2$ with $z_1 - u \geq 0$, then the resulting graph is connected. Moreover if we orient the edges of this graph according to a monomial term order (in the sense of Gröbner bases theory) the directed graph will have a unique sink. Thus from any lattice point of P there is an"augmenting" path to this sink. Thus we have a connected rooted directed tree that can be used very effectively in conjunction with the *reverse-search* enumeration algorithm [6] to obtain an output-sensitive algorithm of constant memory for listing all lattice points of a polyhedron.

Other methods make a much stronger use of complex analysis and convexity (see [11, 8, 40, 59, 62, 63, 73, 84, 75] and the many references within). In a way these algorithms consider counting as a special case of the problem of *integration*. For this recall $\phi_A(b) = \#\{x : Ax = b, x \geq 0, \text{ integer}\}$. If we denote by A_i the columns of the matrix A, then it is easy to see that

$$\sum \phi_A(b) z^b = \frac{1}{\prod_{j=1}^{n} (1 - z^{A_j})}.$$

It is also fairly easy to see that:

$$\sum_{b \in \mathrm{cone}(A) \cap \mathbb{Z}^n} \phi_A(b) e^{-\langle b, z\rangle} = \frac{1}{\prod_{A_i \in A} 1 - e^{-\langle A_i, z\rangle}}.$$

How to invert this equations to recover ϕ_A? Some authors have taken a complex analytic view. For instance, in [17] the integral is evaluated by repeated application of Cauchy's integral theorem and clever manipulations.

$$\phi_A(b) = \frac{1}{(2\pi i)^m} \int_{|z_1| = \epsilon_1} \cdots \int_{|z_m| = \epsilon_m} \frac{z_1^{-b_1 - 1} \cdots z_m^{-b_m - 1}}{(1 - z^{A_1}) \cdots (1 - z^{A_d})}\, dz.$$

Here $0 < \epsilon_1, \dots, \epsilon_m < 1$ are different numbers such that we can expand all the $\frac{1}{1 - z^{M_k}}$ into the power series about 0. This method has been applied very successfully to determine the volumes of Birkhoff polytopes in [16]. The reader knows from basic calculus that to integrate rational functions it is sometimes useful to use a partial fraction decomposition, in order to simplify the expressions. M. Vergne and collaborators have exploited the structure of the hyperplane arrangement associated to the columns of the matrix A in order to have such a simpler decomposition. See [8, 20, 84] and references within for more details and implementation.

We now try to compile a list of software packages for counting lattice points in polyhedra. For 0-1 problems `Azove` is C++ code developed by Markus Behle at the Max Planck Institute of Saarbrucken www.mpi-inf.mpg.de/~behle/azove.html. Lisonek created a `Maple` package that works on knapsack type problems [65]. The Beck-Pixton software [16] can perform fast computations with two-dimensional transportation polytopes. They have the best computation times for that family of polytopes. Vergne and collaborators (see [8]) have developed a package special for unimodular matrices which is very fast. Working with general polytopes we have two programs: one is `ehrhart`, developed by Clauss, Loechner and Wilde (http://icps.u-strasbg.fr/Ehrhart/program/). The first implementation of Barvinok's algorithm was the software `LattE` developed at UC Davis [33, 34]. `LattE` counts lattice points, generates Ehrhart's quasipolynomials, and even solves small integer programs. `LattE` is freely available at www.math.ucdavis.edu/~latte. More recently M. Köppe improved `LattE` and implemented several new algorithms in `LattE macchiato`. It is available at the same web site as LattE or from www.math.ucdavis.edu/~mkoeppe/latte/ (see also [60]). S. Verdoolage wrote a second implementation of the same algorithm, `barvinok` [88]. We now conclude this first section with a more detailed description of Barvinok's algorithm.

1.1 *Counting through Barvinok's Generating Functions*

In the 1990's, based on work by the geometers Brion, Khovanski, Lawrence, and Pukhlikov, A. Barvinok created an algorithm to *count* integer points inside polyhedra. When the dimension is fixed the algorithm can count the number of lattice points in a polytope in polynomial time in the size of the input (the size is given by the binary encoding of the data). See [10, 11] and the references within. Barvinok's result is in a way a generalization of H. Lenstra's algorithm to *detect* integer points in polyhedra. Lenstra's algorithm is based on the LLL-algorithm and the idea of short vectors (see [48, 61]). Shortly after Barvinok's breakthrough, Dyer and Kannan [41] modified the original algorithm of Barvinok, which originally relied on Lenstra's result, giving a new proof that integer programming problems with a fixed number of variables can be solved in polynomial time.

The key ideas of Barvinok's algorithm are using rational functions as an efficient data structure to encode lattice points and a clever way to decompose polyhedra into cones. Given a convex polyhedron P (not necessarily a polytope anymore!), we wish to compute the multivariate generating function

$$f(P; x) = \sum_{\alpha \in P \cap \mathbb{Z}^d} x^\alpha,$$

where $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_d^{\alpha_d}$. This is an infinite formal power series if P is not bounded, but if P is a polytope it is a (Laurent) polynomial with one monomial per lattice point. For example, if we consider a rectangle with vertices $V_1 = (0,0)$, $V_2 = (5000, 0)$, $V_3 = (0, 5000)$, and $V_4 = (5000, 5000)$ the generating function $f(P)$ has over 25,000,000 monomials,

$$f(P, z_1, z_2) = 1 + z_1 + z_2 + z_1 z_2^2 + z_1^2 z_2 + \cdots + z_1^{5000} z_2^{5000}.$$

The representation of $f(P; z_1, z_2)$ as monomials is clearly way too long to be of practical use. But Barvinok's method instead rewrites

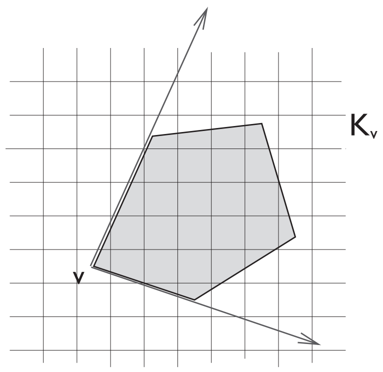

Figure 1. *The tangent cone at vertex v*

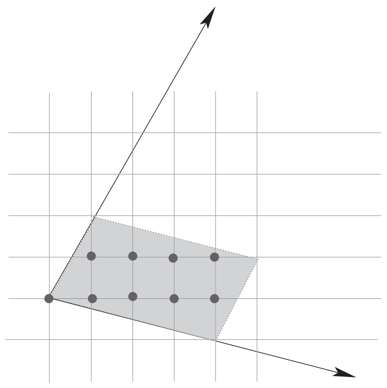

Figure 2. *A two dimensional cone (its vertex at the origin) and its fundamental parallelepiped*

it as a compact sum of rational functions. For instance, only four rational functions suffice to represent the over 25 million monomials. Indeed $f(P, z_1, z_2)$ equals

$$\frac{1}{(1-z_1)(1-z_2)} + \frac{{z_1}^{5000}}{(1-{z_1}^{-1})(1-z_2)} + \frac{{z_2}^{5000}}{(1-{z_2}^{-1})(1-z_1)} + \frac{{z_1}^{5000}{z_2}^{5000}}{(1-{z_1}^{-1})(1-{z_2}^{-1})}.$$

Note that if we wish to know the number of lattice points in P, and we knew $f(P;z)$, we could compute it as the limit when the vector $(x_1,\dots,x_n)$ goes to $(1,1,\dots,1)$. Similarly the maximum of a linear functional over the lattice points equals the highest degree of the univariate polynomial one gets after doing a *monomial substitution* $x_i \to t^{c_i}$ (See [12]). These two calculations can be difficult because of the poles of the rational functions. One uses complex analysis (residue calculations) to find the answer.

A beautiful theorem of M. Brion [19] says that to compute the rational function representation of $f(P;z)$ it is enough to compute the tangent cones at each vertex of P. Let P be a convex polytope and let $V(P)$ be the vertex set of P. Let K_v be the tangent cone at $v \in V(P)$. This is the (possibly translated) cone defined by the facets touching vertex v (see Figure 1). Then the following nice formula holds:

$$f(P;z) = \sum_{v \in V(P)} f(K_v;z).$$

Since it is enough to compute everything for cones, we first explain how to compute the rational function for the "smallest" cones, i.e., the simple cones. A cone is said to be *simple* if its rays are linearly independent vectors. For instance all the tangent cones of the pentagon of Figure 1 are simple. Obtaining the rational function representation of the lattice points of a simple cone $K \subset \mathbb{R}^d$, is easy (see all details in [80] or in Chapter IV of [81]). Here is the formula that one can write directly from the coordinates of the rays of the cone and its *fundamental parallelepiped* Π:

$$f(K;z) = \frac{\sum_{u \in \Pi \cap \mathbb{Z}^d} z^u}{(1-z^{c_1})(1-z^{c_2})\dots(1-z^{c_d})}.$$

Here Π is the half open parallelepiped $\{x : x = \alpha_1 c_1 + \cdots + \alpha_d c_d,\ 0 \le \alpha_i < 1\}$. A two-dimensional example is shown in Figure 2: we have $d = 2$ and $c_1 = (1,2)$, $c_2 = (4,-1)$. We have:

$$f(K;z) = \frac{z_1^4 z_2 + z_1^3 z_2 + z_1^2 z_2 + z_1 z_2 + z_1^4 + z_1^3 + z_1^2 + z_1 + 1}{(1-z_1 z_2^2)(1-z_1^4 z_2^{-1})}.$$

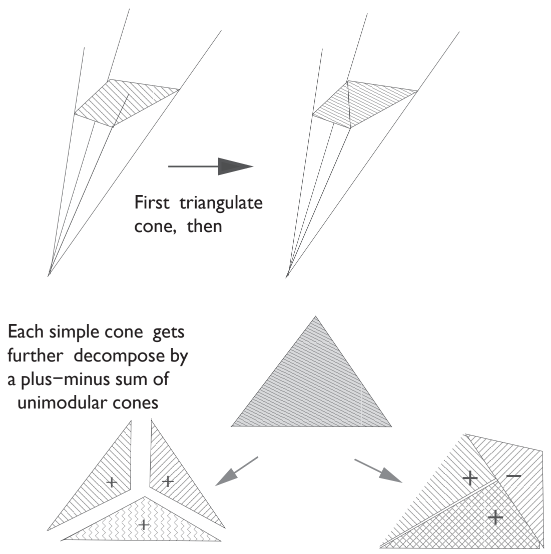

Figure 3. *The signed decomposition of a cone into unimodular cones. Two general steps*

But what to do if the cone K is not simple? Break it into simple cones! The wonderful idea of A. Barvinok was observing that, although triangulating the cone K may be an inefficient way to subdivide the cone (i.e., there are exponentially many pieces), if one is willing to add *and* substract cones for fix dimension d, there exists a polynomial time algorithm which decomposes a rational polyhedral cone $K \subset \mathbb{R}^d$ into simple unimodular cones K_i. A simple cone is *unimodular* if its fundamental parallelepiped is a single lattice point. See Figure 3. In fact, via the decomposition, with numbers $\epsilon_i \in \{-1, 1\}$ we can write an expression

$$f(K;z) = \sum_{i \in I} \epsilon_i f(K_i;z), \quad |I| < \infty.$$

The index set I is of size polynomial in the input data as long as we work in fixed dimension.

2 Estimation and Approximate Counting

Since exact count is hard it is not surprising that there is a rich and exciting theory of estimation and approximation. Now we present various algorithms applicable to arbitrary polyhedra (much work has also been done for special families, such as contingency tables, see for instance [8, 14, 16, 32, 38, 72] and the many references there).

Unfortunately we cannot avoid omitting part of this interesting literature. We recommend the surveys [43, 54, 53, 89] and the references therein for more information. One important point in the theory of approximate counting was that in 1986 Jerrum, Valiant and Vazirani showed that the existence of a good approximation (i.e., a fully polynomial randomized approximation scheme) is equivalent to almost uniform sampling from the set of lattice points [55]. Thus with the close connection to sampling the effort has often been on finding reasonable sampling procedures.

One approach is *direct Monte-Carlo sampling* or *dart-throwing techniques*. The basic framework is that we wish to estimate the size $|S|$ of the lattice point set S of a given polyhedron. To do this, we enclose S in an easy set S' which approximates S and can be sampled uniformly (think of S' a nice box around a nasty looking S). The process is then to sample from S' to get an estimate $\hat{p}$ of the proportion $|S|/|S'|$ of sampled points which lie in S. Estimate $|S|$ by $\hat{p}|S'|$. Of course, the key to using this approach is to find a set S' such that $S \subset S'$, $|S'|$ is easy to determine and can be easily sampled. Finally, it is desirable that $|S'|/|S|$ is polynomially bounded. There are several clever variations of this simple idea. One noteworthy example is the combination of dart-throwing with dynamic programming which was proposed by Dyer in [42]. Another example is the use of randomized rounding to improve the dart-throwing technique [58].

Another approach is the *Markov chain Monte-Carlo method* (MCMC) in which we try to sample using a random walk directly in the set of lattice points and hopefully proving that this will approach the uniform distribution in time polynomial in the instance size (so called *rapid mixing*). Notable successes of MCMC include volume computation, permanents, $0/1$-knapsacks, (see [54, 74] and the references therein). In special polytopes (e.g., contingency tables) it is not a problem to design a random walk, thus the main difficulty of MCMC is in proving the polynomial time convergence property to a uniform distribution. This means that the method is not as easy to apply rigorously as it may appear at first sight. On the other hand, for arbitrary polyhedra, e.g., in the context of integer programming where we do not even know whether the polytope has lattice points at all, it is not easy to automatically generate a well-behaved Markov chain. The algebraic approach in Section 1 helps with this. The paper [39] provides a way to generate a Markov chain as a walk on the lattice point graph defined by the toric ideal. Unfortunately, Markov chain moves (a *Markov basis*) can be quite large and hard to compute (see, e.g., [37]).

These difficulties motivate a search for other ways to sample lattice points. In the context of two-way contingency tables [26] introduced an alternative to the MCMC method. *Sequential importance sampling* uses the idea of divide and conquer to sequentially build up the proposed distribution by sampling variable entries one by one. Roughly speaking, the value of a particular solution for $Ax = b,\ x \geq 0$ can be done entry by entry, where x_i is constrained by the choices made for $x_1, \dots, x_{i-1}$. The sampling is easy to do if geometrically the possible solutions for x_i form an interval of values without gaps. In [27] many real examples show that sequential importance sampling is applicable and useful, and can be used in cases where a Markov basis cannot be easily computed (e.g., multiway contingency tables). This method had great success on the problem of estimating the number of labeled graphs with a prescribed degree sequence (see [23]).

Lately, a strong connection between convex optimization and integer feasibility and counting has been observed by several authors [52, 70, 71, 15]. Most recently Barvinok and Hartigan [15] presented probabilistic theorems for the estimation of the number of lattice points or 0-1 points inside a polyhedron which are based on convex programming tools. To conclude this section, we take a closer look at this very recent development.

2.1 Estimation through Convex Optimization

We now briefly recount the main results presented in [15] that we used here to compute estimations on the number of lattice points inside a polyhedron by solving specially constructed convex optimization problems on polytopes. Unlike other techniques the Barvinok-Hartigan idea uses no prior knowledge about the input polyhedron. In what follows, a *polytope* $P \subset \mathbb{R}^n$ is the set of solutions to the system $Ax = b, x \geq 0$ where $A = [a_1, \dots, a_n]$ is the matrix with d-dimensional vectors $a_1, \dots, a_n$ and x is thought of as a column vector $x = [\xi_1, \dots, \xi_n]^T$. For simplicity we assume that vectors $a_1, \dots, a_n$ span $\mathbb{R}^d$ and that in fact the vectors $a_1, \dots, a_n$ and b are integral, that is, $a_1, \dots, a_n; b \in \mathbb{Z}^d$. We also assume that P has a non-empty interior, that is, contains a point $x = (\xi_1, \dots, \xi_n)$, where the defining inequalities are strict (it is well-known this can be checked fast both in theory and in practice). We wish to estimate the number $|P \cap \mathbb{Z}^n|$ of integer points in P. We also consider the problem of counting the set of all vectors in P with the coordinates 0 and 1.

Recall that a discrete random variable x has geometric distribution if $\mathbf{Pr}\{x = k\} = pq^k$ for $k = 0, 1, \dots$. for positive numbers p and q such that $p + q = 1$. For the expectation and variance of x we have $\mathbf{E}\, x = \frac{q}{p}$ and $\mathbf{var}\, x = \frac{q}{p^2}$ respectively. Conversely, if $\mathbf{E}\, x = \zeta$ for some $\zeta > 0$ then $p = \frac{1}{1+\zeta}$, $q = \frac{\zeta}{1+\zeta}$ and $\mathbf{var}\, x = \zeta^2 + \zeta$.

Theorem 1 (Theorem 2.1 in [15]). *Let $P \subset \mathbb{R}^n$ be the intersection of an affine subspace in $\mathbb{R}^n$ and the non-negative orthant $\mathbb{R}^n_+$. Suppose that P is bounded and has a non-empty interior, that is contains a point $y = (\eta_1, \dots, \eta_n)$ where $\eta_j > 0$ for $j = 1, \dots, n$.*

Then the strictly concave function

$$g(x) = \sum_{j=1}^{n} \left(\left(\xi_j + 1\right) \ln \left(\xi_j + 1\right) - \xi_j \ln \xi_j \right)$$

attains its maximum value on P at a unique point $z = (\zeta_1, \dots, \zeta_n)$ such that $\zeta_j > 0$ for $j = 1, \dots, n$.

Suppose now that x_j are independent geometric random variables with expectations ζ_j for $j = 1, \dots, n$. Let $X = (x_1, \dots, x_n)$. Then the probability mass function of X is constant on $P \cap \mathbb{Z}^n$ and equal to $e^{-g(z)}$ at every $x \in P \cap \mathbb{Z}^n$. In particular,

$$\mathbf{Pr}\{X \in P\} = e^{-g(z)} \left|P \cap \mathbb{Z}^n\right|.$$

Similarly, Barvinok and Hartigan proposed a clever probability mass distribution which is constant on the 0-1 points. Let p and q be positive numbers such that $p + q = 1$. We recall that a discrete random variable x has Bernoulli distribution if $\mathbf{Pr}\{x = 0\} = p$ and $\mathbf{Pr}\{x = 1\} = q$. We have $\mathbf{E}\, x = q$ and $\mathbf{var}\, x = qp$. Conversely, if $\mathbf{E}\, x = \zeta$ for some $0 < \zeta < 1$ then $p = 1 - \zeta$, $q = \zeta$ and $\mathbf{var}\, x = \zeta - \zeta^2$. Their second main result is as follows.

Theorem 2 (Theorem 2.4 in [15]). *Let $P \subset \mathbb{R}^n$ be the intersection of an affine subspace in $\mathbb{R}^n$ and the unit cube $\{0 \leq \xi_j \leq 1 : \ j = 1, \dots, n\}$. Suppose that P has a non-empty interior, that is, contains a point $y = (\eta_1, \dots, \eta_n)$ where $0 < \eta_j < 1$ for $j = 1, \dots, n$. Then the strictly concave function*

$$h(x) = \sum_{j=1}^{n} \left(\xi_j \ln \frac{1}{\xi_j} + \left(1 - \xi_j\right) \ln \frac{1}{1 - \xi_j} \right)$$

attains its maximum value on P at a unique point $z = (\zeta_1, \dots, \zeta_n)$ such that $0 < \zeta_j < 1$ for $j = 1, \dots, n$.

Suppose now that x_j are independent Bernoulli random variables with expectations ζ_j for $j = 1, \dots, n$. Let $X = (x_1, \dots, x_n)$. Then the probability mass function of X is constant on $P \cap \{0, 1\}^n$ and equal to $e^{-h(z)}$

for every $x \in P \cap \{0,1\}^n$. In particular,

$$\mathbf{Pr}\{X \in P\} = e^{-h(z)}\left|P \cap \{0,1\}^n\right|.$$

Now we discuss the important practical consequences of the above results. First of all, clearly the two theorems above suggest two simple sampling algorithms

1. Solve the optimization problem
   $\gamma(P) = \max \sum_{j=1}^n g(x_j)$
   subject to $x \in P$.
2. Let $z = (\zeta_1, \ldots, \zeta_n)$ be the unique optimal point and γ the optimum value. Sample a random point $X = (x_1, \ldots, x_n)$ by sampling x_j independently at random from the geometric distribution with expectation ζ_j, that is with parameter $p_j = (1+\zeta_j)^{-1}$. Check whether $X \in P$. Repeat many times and compute the proportion α of times that the point X lands in P (i.e., $0 \le \alpha \le 1$).
3. Estimate $|P \cap \mathbb{Z}^n| \approx \alpha e^{\gamma}$.

Similarly, by using the entropy function h of Theorem 2, instead of the concave function g of Theorem 1, we can get a sampling algorithm to estimate 0-1 points inside P:

1. Solve the optimization problem
   $\gamma(P) = \max \sum_{j=1}^n h(x_j)$
   subject to $x \in P$.
2. Let $z = (\zeta_1, \ldots, \zeta_n)$ be the unique optimal point and γ the optimum value. Sample a random binary point $X = (x_1, \ldots, x_n)$ by sampling x_j independently at random from the Bernoulli distribution where x_i is equal to 1 with probability ζ_j.
3. Check whether $X \in P$. Repeat many times and compute the proportion α of times that the point X lands in P (i.e., $0 \le \alpha \le 1$).
4. Estimate $|P \cap \{0,1\}^n| \approx \alpha e^{\gamma}$.

At the same time, Barvinok and Hartigan developed a heuristic approximation to the above numbers that does not require sampling. We describe it first for general integer programs. For a polytope P, defined by a system $Ax = b, x \ge 0$, we find the point $z = (\zeta_1, \ldots, \zeta_n)$ maximizing

$$g(x) = \sum_{j=1}^{n}\left(\left(\xi_j + 1\right)\ln\left(\xi_j + 1\right) - \xi_j \ln \xi_j\right)$$

on P. Let $A = \left(\alpha_{ij}\right)$. We compute the $d \times d$ matrix $Q = \left(q_{ij}\right)$ by

$$q_{ij} = \sum_{k=1}^{n} \alpha_{ik}\alpha_{jk}\left(\zeta_k^2 + \zeta_k\right).$$

We assume that A is an an integer $n \times d$ matrix of rank $d < n$. Let $\Lambda = A(\mathbb{Z}^n)$ be image of the standard lattice, $\Lambda \subset \mathbb{Z}^d$. The proposed approximation the number of integer points in P is given by

$$\left|P \cap \mathbb{Z}^n\right| \approx \frac{\det \Lambda}{(2\pi)^{d/2}(\det Q)^{1/2}}.$$

Similarly, for estimating 0-1 points inside P, we find the point $z = (\zeta_1, \ldots, \zeta_n)$ maximizing

$$h(x) = \sum_{j=1}^{n}\left(\xi_j \ln \frac{1}{\xi_j} + (1-\xi_j)\ln \frac{1}{1-\xi_j}\right)$$

on the truncated polytope defined by the equations $Ax = b$ and inequalities $0 \le x \le 1$ (that is, $0 \le \xi_j \le 1$ for $j = 1, \ldots, n$). We then compute the $d \times d$ matrix $Q = \left(q_{ij}\right)$ by

$$q_{ij} = \sum_{k=1}^{n} \alpha_{ik}\alpha_{jk}\left(\zeta_k - \zeta_k^2\right).$$

This way the desired approximation to the number of 0-1 vectors in P is given by

$$\left|P \cap \{0,1\}^n\right| \approx \frac{\det \Lambda}{(2\pi)^{d/2}(\det Q)^{1/2}},$$

where $\Lambda = A(\mathbb{Z}^n)$.

Remark. It is well-known that all the above mentioned optimization problems can be solved quite efficiently, both in theory and in practice, by interior point methods, see [22, 77]. This is an additional attraction of these ideas.

To conclude we report on the practical experimental performance of the two estimation algorithms derived from the theory in [15]. Our test sets included Knapsacks (both general and binary, some randomly generated and others artificially constructed), multiway transportation problems, market split problems, and a combinatorial 0-1 problems (graph enumeration). In all instances, the goal was to measure the accuracy of the estimation. We observed that, in a good number of cases, the Barvinok-Hartigan techniques perform rather well. Most important, the methods used here require no prior knowledge of the polyhedron (e.g., integral feasibility) and work for arbitrary general rational polyhedra. We divided our experiments into four families: (1) knapsack simplices, (2) flow polytopes and two and three-way contingency tables, (3) market split problem, and (4) combinatorial $0-1$ polyhedra. Each category is reported in the Appendix. Note that categories (1) and (2) are polyhedra with lattice points with "large" integer entries, while categories (3) and (4) are testing examples of binary polyhedra. In all cases the problems are given in the canonical form $Ax = b, x \ge 0$ and we checked for accuracy of the estimation using the software `LattE` and Azove.

All computations were done on a 2 GHz Pentium PC running Red Hat Linux. For the solution of the necessary convex optimization problems we used the software `LOQO` a software package for solving general (smooth) nonlinear optimization problems. `LOQO` implements an infeasible-primal-dual path-following method. Even though `LOQO` can handle nonconvex problems in general, it performs much better with convex problems (see [21, 86, 87] and references therein). This was indeed the case in our situation and in absolutely all the instances that we tested the number of iterations was never above twenty. Thus we can say that finding the necessary estimation parameters can be done easily and efficiently.

The simulation of the geometric and the Bernoulli distributions used in the two sampling algorithms was done using the statistics package of `MAPLE` version 12. Each experiment consisted on finding the parameters γ and the probability distribution numbers for each instance. Then we sampled three different times with (1000), (5000), and (10000) samples and the same test parameters read from `LOQO` and we report on the average. As described in the algorithm, the sampling parameters are the exponential exponent γ and the parameters for geometric and Bernoulli random variables which come in the form of a point of the polytope.

The evidence we have presented in the Appendix supports the usefulness and accuracy of the Barvinok-Hartigan Gaussian estimation. The easy-to-use Gaussian estimation is widely applicable and fairly accurate in most instances we tried. In 40 percent of tests done

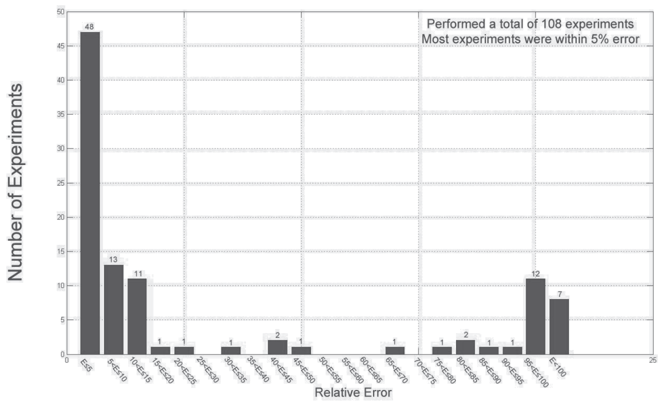

Figure 4. *Error estimation over tests*

using the Gaussian estimation the relative error of the Gaussian estimation was 5% or less. Only in the most pathological cases (e.g., market split, Aardal knapsacks) one can see a prediction worse than twice the correct value. See Figure 4 for details. We noted also that the algorithms seem to scale well as the number of variables grows. The sampling algorithm is nevertheless not very good. It performed best for knapsack problems but when two or more constraints are in play it is clearly not as reliable for estimating the number of lattice points in polyhedra. Using aggregation of constraints did not help with the problem.

Of course, as expected from the theoretical hardness of the problem, one can indeed find rather nasty problems for which the number of lattice points grows arbitrarily large and yet the estimation is consistently poor. Nevertheless, the experiments actually suggest that the convex optimization problems solved for finding the sampling parameters could in principle be used as a fairly good measure of how "well-behaved" is an integer linear program will turn out to be.

We conclude by saying that although other algorithms are available to carry out the same kind of estimation, we are not aware of any other algorithm that works in such a generality without prior knowledge of the integer feasibility of the polyhedron in question. For instance, the MCMC and Importance Sampling method both rely on the ability of generating random lattice points according to a pivot or completion procedures. This is unavailable in most polyhedra. Thus the Barvinok-Hartigan process has a distinctive advantage of not requiring prior knowledge about the input polyhedron.

Acknowledgements

The author is grateful to the Institute for Pure and Applied Mathematics at UCLA for their hospitality during the time this article was written. The author gratefully acknowledges support from NSF grant DMS-0914107. He also wants to thank Prof. Alexander Barvinok, Prof. Benjamin Morris, and Prof. Hande Benson for many useful conversations that greatly helped the creation of this article. Very special thanks go to Prof. Andrea Lodi for his help and support.

Appendix

Details for experiments with the Gaussian estimation of Barvinok and Hartigan are available in the online-appendix, accessible at www.mathprog.org//Optima-Issues/optim81-app.pdf.

J. A. De Loera, Department of Mathematics, University of California, Davis CA, USA. deloera@math.ucdavis.edu, www.math.ucdavis.edu/~deloera

References

[1] Aardal, K., Lenstra, A.K., and Lenstra, H.W. Jr.: *Hard equality constrained integer knapsacks*. In: W.J. Cook and A.S. Schulz (eds.), Integer Programming and Combinatorial Optimization: 9th International IPCO Conference, Lecture Notes in Computer Science vol. 2337, Springer-Verlag, 350–366, 2002.

[2] Aardal, K., Weismantel, R., and Wolsey, L.: *Non-standard approaches to integer programming* Discrete Applied Math. 123 No. 1–3, (2002), 5–74.

[3] Achterberg T., Heinz S., and Koch T. *Counting solutions of integer programs using unrestricted subtree detection* ZIB technical report 08–09, 2008.

[4] Adleman, L. and Manders, K.: *NP-complete decision problems for quadratic polynomials*. In: Proceedings of the Eighth annual ACM symposium on theory of computing, Hershey PA 1976, 23–29, ACM, New York, 1976.

[5] Andrews, G., Ekhad, S., and Zeilberger, D.: *A short proof of Jacobi's formula for the number of representations of an integer as a sum of four squares*, American Math. Monthly, 100, 274–276, 1993.

[6] Avis, D. and Fukuda, K, *Reverse Search for Enumeration*, Discrete Applied Math, Vol. 6, 21–46 (1996)

[7] Balcioglu, A. and Wood R.: *Enumerating Near-Min s-t Cuts*. In: D.L. Woodruff (editor) Network Interdiction and Stochastic Integer Programming , vol. 22, series operations research/computer science interfaces, Kluwer, Boston 2002.

[8] Baldoni-Silva, W., De Loera, J., and Vergne, M.: *Counting integral flows on Networks*, Foundations of Computational Mathematics, Vol. 4, 277–314, 2004.

[9] Barvinok, A.I.: *A course in Convexity*, Graduate studies in Mathematics, vol. 54, Amer. Math. Soc., Providence RI, 2002.

[10] Barvinok, A.I.: *Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math of Operations Research, Vol. 19, 769–779, 1994.

[11] Barvinok, A.I. and Pommersheim, J.: *An algorithmic theory of lattice points in polyhedra*. In: New Perspectives in Algebraic Combinatorics (Berkeley, CA, 1996–1997), 91–147, Math. Sci. Res. Inst. Publ. 38, Cambridge Univ. Press, Cambridge, 1999.

[12] Barvinok, A.I. and Woods, K.: *Short rational generating functions for lattice point problems*, J. Amer. Math. Soc., Vol. 16, 957–979, 2003.

[13] Barvinok, A.I., *Enumerating contingency tables via random permanents*, Combinatorics, Probability and Computing, v.17 n.1, (2008) 1–19.

[14] Barvinok, A.I, *Asymptotic estimates for the number of contingency tables, integer flows, and volumes of transportation polytopes*, International Mathematics Research Notices 2009 (2009), 348Ð385.

[15] Barvinok, A. I. and Hartigan, J. *Maximum entropy Gaussian approximation for the number of integer points and volumes of polytopes* manuscript 2009, available at www.math.lsa.umich.edu/~barvinok/papers.html and at the Arvix http://arxiv.org/abs/0903.5223

[16] Beck, M. and Pixton, D., *The Ehrhart polynomial of the Birkhoff polytope,*. Discrete Comput. Geom. 30, no. 4 (2003), 623Ð637.

[17] Beck, M.: *Counting lattice points by means of the residue theorem*. Ramanujan Journal, 4, no. 3, 299–310, 2000.

[18] Behle, M. and Eisenbrand, F. *0-1 vertex and facet enumeration with BDDs* In David Applegate et al. (Eds.), Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX'07), New Orleans, U.S.A., SIAM, p. 158–165, 2007.

[19] Brion, M.: *Points entiers dans les polyèdres convexes. Ann. Sci. École Norm. Sup.*, Vol. 21, 653–663, 1988.

[20] Brion, M. and Vergne, M.: *Residue formulae, vector partition functions and lattice points in rational polytopes*, J. Amer. Math. Soc., Vol. 10, 797–833, 1997.

[21] Benson H., Shanno, D.F. and Vanderbei, R.J. *A Comparative Study of Large Scale Nonlinear Optimization Algorithms* In High Performance Algorithms and Software for Nonlinear Optimization, (G. Di Pillo and A. Murli, editors), 94–126, Kluwer Academic Publishers, 2003.

[22] Ben-Tal, A. and Nemirovski, A. *Lectures on modern convex optimization: analysis, algorithms, and engineering applications*, MPS-SIAM series on Optimization, 2001.

[23] Blitzstein, J. and P. Diaconis A sequential importance sampling algorithm for generating random graphs with prescribed degrees, manuscript (2009).

[24] M.R.Bussieck and M.E.Lübbecke *The vertex set of a 0/1-polytope is strongly P-enumerable* Comput. Geom., 11(2):103–109, 1998.

[25] Chen, Y, I. Dinwoodie, A. Dobra, and M. Huber, *Lattice Points, Contingency Tables, and Sampling.* Contemporary Mathematics, Vol. 374, 65–78. American Mathematical Society, (2005).

[26] Chen, Y., Diaconis, P., Holmes, S., and Liu, J. S. *Sequential Monte Carlo methods for statistical analysis of tables*. Journal of the American Statistical Association, 100, (2005) 109–120.

[27] Chen, Y., Dinwoodie, I. H., and Sullivant, S. *Sequential importance sampling for multiway tables*. The Annals of Statistics, 34, (2006) 523-545.

[28] Clauss, P. and Loechner, V.: *Parametric Analysis of Polyhedral Iteration Spaces*, Journal of VLSI Signal Processing, Vol. 19, 179–194, 1998.

[29] Cornuejols, G. and Dawande, M. *A class of hard small 0-1 programs*. In R. E. Bixby, E.A. Boyd, and R.Z. Rios-Mercado, editors, Integer Programming and Combinatorial Optimization. 6th International IPCO Conference, Houston, Texas, June 1998. Springer Lecture Notes in Computer Science 1412, Heidelberg, 1998.

[30] Cox, D., Little, J., and O'Shea D. *Ideals, varieties and algorithms*, Undergraduate Texts in Mathematics, Springer, New York, 1992.

[31] Danna E., Fenelon M., Gu Z., and Wunderling R.: *Generating multiple solutions for mixed integer programming problems* In M. Fischetti and D.P. Williamson, eds., Integer Programming and Combinatorial Optimization, 12th International IPCO conference, Ithaca New York, June 2007. Springer Lecture Notes in Computer Science 4513, 280–294.

[32] De Loera, J. and Sturmfels, B. *Algebraic unimodular counting*, Math. Programming Ser. B., Vol. 96, No. 2, (2003), 183–203.

[33] De Loera, J. A., Haws, D., Hemmecke, R., Huggins, P., Tauzer, J. and Yoshida, R.: *A User's Guide for* `LattE`, software package `LattE` and manual are available at www.math.ucdavis.edu/~latte/ 2003.

[34] De Loera, J. A. and Hemmecke, R. and Tauzer, J. and Yoshida, R.: *Effective lattice point counting in rational convex polytopes*, J. Symbolic Comp., vol. 38, 1273–1302, 2004.

[35] De Loera, J.A, Hemmecke R., Köppe M., and Weismantel R. Integer polynomial optimization in fixed dimension. *Mathematics of Operations Research.* (2006), vol 31, No. 1, 147–153.

[36] De Loera, J.A, Hemmecke R. and Köppe M. Pareto optima of multicriteria integer programs, *INFORMS journal of Computing.* Vol. 21, No. 1, (2009), 39–48.

[37] De Loera, J. A. and Onn, S. *Markov bases of three-way tables are arbitrarily complicated* J. Symbolic Computation, 41:173–181, 2006.

[38] Diaconis, P. and Gangolli, A.: *Rectangular arrays with fixed margins,* In: Discrete probability and algorithms (Minneapolis, MN, 1993), 15–41, IMA Vol. Math. Appl., 72, Springer, New York, 1995.

[39] Diaconis, P. and Sturmfels, B. *Algebraic Algorithms for Sampling from Conditional Distributions*. Annals of Statistics, 26(1): 363–397, 1998.

[40] Diaz, R. and Robins, S.: *The Ehrhart Polynomial of a Lattice Polytope*, Annals of Mathematics, 145, 503–518, 1997.

[41] Dyer, M. and Kannan, R. *On Barvinok's algorithm for counting lattice points in fixed dimension*. Math of Operations Research 22 (1997) 545–549.

[42] Dyer, M. *Approximate counting by dynamic programming* Proceedings of the 35th Annual ACM Symposium on the Theory of Computing (STOC 03), ACM Press, pp. 693–699, 2003.

[43] Dyer, M., Mount, D., Kannan, R. and Mount, J.: *Sampling contingency tables,* Random Structures and Algorithms, Vol. 10, 487–506, 1997.

[44] Ehrhart, E.: *Polynômes arithmétiques et méthode des polyèdres en combinatoire*, International Series of Numerical Mathematics, Vol. 35, Birkhäuser Verlag, Basel, 1977.

[45] Fienberg, S.E., Makov, U.E., Meyer, M.M. and Steele, R.J.: *Computing the exact conditional distribution for a multi-way contingency table conditional on its marginal totals.* In: *Data Analysis From Statistical Foundations*, 145–166, Nova Science Publishers, A. K. Md. E. Saleh, Huntington, NY, 2001.

[46] Garey, M.R. and Johnson, S.J.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.

[47] Ghosh, S., Martonosi, M., Malik, S.: *Cache miss equations: a compiler framework for analyzing and tuning memory behavior,* ACM transactions on programming languages and systems, 21, 703–746, 1999.

[48] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization*, second edition. Algorithms and Combinatorics, 2, Springer-Verlag, Berlin, 1993.

[49] Guy, R. K. *Unsolved Problems in Number Theory,* 2nd ed. New York: Springer-Verlag, 240–241, 1994.

[50] Huxley, M.N. *Area, lattice points, and exponential sums*, Oxford Univ. Press. Oxford, 1996.

[51] Jacobi, C. Gesammelte Werke, Berlin 1881–1891. Reprinted by Chelsea, New York, 1969.

[52] Jarre, F. *Relating max-cut problems and binary linear feasibility problems* available at www.optimization-online.org/DB_FILE/2009/02/2237.pdf

[53] Jerrum, M.R. *Counting, sampling and integrating: algorithms and complexity*, Birkhäuser, Basel, 2003.

[54] Jerrum, M.R. and Sinclair, A.: *The Markov Chain Monte Carlo Method: An approach to approximate Counting and integration*, in: Dorit Hochbaum (Ed.), Approximation Algorithms, PWS Publishing Company, Boston, 1997.

[55] Jerrum, M.R., Valiant, L.G., and Vazirani, V.V.. *Random generation of combinatorial structures from a uniform distribution*, Theoretical Computer Science, 43:169Ð188, 1986.

[56] Jerrum M., Sinclair A., and Vigoda, E., *A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries.* Journal of the ACM, 51(4):671-697, 2004.

[57] Jones, J.P.: *Universal diophantine equations,* Journal of symbolic logic, 47 (3), 403–410, 1982.

[58] Kannan, R. and Vempala, S. *Sampling lattice points* in proceedings of the twenty-ninth annual ACM symposium on Theory of computing El Paso, Texas USA, 1997, 696–700 .

[59] Kantor, J.M. and Khovanskii, A.: *Une application du Théorème de Riemann-Roch combinatoire au polynôme d'Ehrhart des polytopes entier*, C. R. Acad. Sci. Paris, Series I, Vol. 317, 501–507, 1993.

[60] Köppe, M. *A primal Barvinok algorithm based on irrational decompositions* To appear in SIAM Journal on Discrete Mathematics.

[61] Lenstra, H.W. *Integer Programming with a fixed number of variables* Mathematics of Operations Research, 8, 538–548

[62] Lasserre, J.B. and Zeron, E.S.: *On counting integral points in a convex rational polytope* Math. Oper. Res. 28, 853–870, 2003.

[63] Lasserre, J.B. and Zeron, E.S *A simple explicit formula for counting lattice points of polyhedra*. Lecture Notes in Computer Science 4513, Springer Verlag pp. 367–381 (2007)

[64] Micciancio, D. and Goldwasser, S. *Complexity of lattice problems. A cryptographic perspective*, Kluwer International Series in Engineering and Computer Science, 671. Kluwer, Boston, MA, 2002.

[65] Lisonek, P.: *Denumerants and their approximations,* Journal of Combinatorial Mathematics and Combinatorial Computing, Vol. 18, 225–232, 1995.

[66] Loebl M., Zdeborova L., *The 3D Dimer and Ising Problems Revisited*, European J. Combinatorics No. 29 Vol. 3, (2008).

[67] Loechner, V., Meister, B., and Clauss, P.: *Precise data locality optimization of nested loops,* J. Supercomput., Vol. 21, 37–76, 2002.

[68] Macdonald, I. G.: *Polynomials associated with finite cell-complexes*, J. London Math. Soc. (2), 4, 181–192, 1971.

[69] Mahadev, N.V.R. and Peled, U.: *Threshold graphs and related topics*. Annals of Discrete Mathematics, Vol 56, North-Holland Publishing Co. Amsterdam.

[70] Mangasarian, O.L: *Knapsack Feasibility as an Absolute Value Equation Solvable by Successive Linear Programming*, Optimization Letters 3(2), (2009), 161–170.

[71] Mangasarian O.L. and Recht, B.: *Probability of unique Integer solution to a system of linear equations* Data Mining Institute, Univ. of Wisconsin,Technical Report 09-01, September 2009.

[72] Mount, J.: *Fast unimodular counting*, Combinatorics, Probability, and Computing 9 (2000) 277–285.

[73] Morelli, R.: *Pick's theorem and the Todd class of a toric variety*, Adv. Math. 100, 183–231, 1993.

[74] Morris, B. and Sinclair A.: *Random walks on truncated cubes and sampling 0-1 knapsack solutions*. SIAM journal on computing, 34 (2004), pp. 195-226

[75] Pemantle, R. and Wilson, M.: *Asymptotics of multivariate sequences, part I: smooth points of the singular variety*. J. Comb. Theory, Series A, vol. 97, 129–161, 2001.

[76] Pesant G.: *Counting and Estimating Lattice Points: Special polytopes for branching heuristics in constraint programming*. This issue.

[77] Renegar, J.: *A mathematical view of interior-point methods in convex optimization* MPS-SIAM series in Optimization, 2001

[78] Sankaranarayanan S, Ivancic, F, and Gupta, A.: *Program analysis using symbolic ranges*, in Proceedings of the Static Analysis Symposium 2007, vol. 4634 of Lecture Notes in Computer Science, Springer, 366–383.

[79] Schrijver, A.: *Theory of Linear and Integer Programming*. Wiley-Interscience, 1986.

[80] Stanley, R. P.: *Decompositions of rational convex polytopes*. In: Combinatorial mathematics, optimal designs and their applications (Proc. Sympos. Combin. Math. and Optimal Design, Colorado State Univ., Fort Collins, Colo., 1978), Ann. Discrete Math., 6, 333–342, 1980.

[81] Stanley, R.P.: *Enumerative Combinatorics*, Volume I, Cambridge, 1997.

[82] Sturmfels, B.: *Gröbner bases and convex polytopes*, university lecture series, vol. 8, AMS, Providence RI, (1996).

[83] Soprunov, I. and Soprunova, J.: *Toric surface codes and Minkowski length of polygons*, SIAM J. Discrete Math. 23 (2008/09), no. 1, 384–400.

[84] Szenes, A. and Vergne, M.: *Residue formulae for vector partitions and Euler-Maclaurin sums*, Adv. in Appl. Math, 30, 295–342, 2003.

[85] Thomas, R.: *Algebraic methods in integer programming*, In Encyclopedia of Optimization (eds: C. Floudas and P. Pardalos), Kluwer Academic Publishers, Dordrecht, 2001.

[86] Vanderbei, R. J.: *LOQO user's manual – version 3.10* Optimization Methods and Software, Vol. 11, Issue 1–4 (1999), 485–514.

[87] Vanderbei, R.J and Shanno, D.F.: *An Interior-Point Algorithm for Nonconvex Nonlinear Programming*. Computational Optimization and Applications, 13: (1999) 231–252, 1999.

[88] Verdoolaege, S. *barvinok*: User guide. Available from URL http://freshmeat.net/projects/barvinok/, 2007

[89] Welsh, D.J.A.: *Approximate counting* in Surveys in Combinatorics, (edited by R. A. Bailey), Cambridge Univ. Press, Cambridge, 1997.

[90] Ziegler, G.M.: *Lectures on Polytopes*, Springer-Verlag, New York, 1995.

Gilles Pesant

Counting and Estimating Lattice Points: Special Polytopes for Branching Heuristics in Constraint Programming

1 Introduction

The Counting Problem in combinatorics is important in many areas and for different reasons (see for example [4] in this issue). The purpose of this paper is to describe recent efforts in using counting to guide heuristic branching within a Constraint Programming framework, in order to find solutions to combinatorial problems. In such a context where counting is done repeatedly and used as a heuristic, exactness is not crucial but execution speed is. The algorithms we will describe often trade some accuracy for faster computation, given that one cannot spend more than a fraction of a second on a given counting instance. What helps is that we will not count solutions to the problem as a whole but to parts of that problem, as will become clear later.

In the rest of this paper we present a short introduction to Constraint Programming and constraint-centered search heuristics. We then concentrate on three families of special polytopes and on how we evaluate the number of lattice points they contain.

2 Constraint Programming

Constraint Programming (CP) is a powerful technique to solve combinatorial problems [17]. It applies sophisticated inference to reduce the search space and a combination of variable- and value-selection heuristics to guide the exploration of that search space. Like Integer Programming (IP), one states a model of the problem at hand in mathematical language and also builds a search tree through problem decomposition. But there are mostly important differences, among them:

- CP works directly on discrete variables instead of relying mostly on a continuous relaxation of the model;
- the modeling language offers many high-level primitives representing common combinatorial substructures of a problem – often a few constraints are sufficient to express complex problems;
- each of these primitives, which from now on we shall call *constraints*, may have its own specific algorithm to help solve the problem;
- one does not branch on fractional variables but instead on indeterminate variables, which currently can take several possible values (variables are not necessarily fixed to a particular value at a node of the search tree);
- even though CP can solve optimization problems, it is primarily designed to handle feasibility problems.

In its most general form, a problem is modeled as

$$\begin{array}{rcll} \mathbf{x} & \in & C_j \subset \mathbb{Z}^n & 1 \le j \le p \\ x_i & \in & D_i \subset \mathbb{Z} & 1 \le i \le n \end{array}$$

where $\mathbf{x}$ is the vector of variables $(x_1, \ldots, x_n)$, each D_i, called the *domain* of x_i, is finite, and each C_j is a constraint on the variables, restricting the combination of values from their respective domains. Constraints are more naturally expressed on a subset of the variables (called its *scope*), and indeed we will most often present them in that way, writing $C(x_1, \ldots, x_m)$ to mean constraint C on variables $x_1, \ldots, x_m$. In any case they can be trivially extended over all the variables to fit this form. The following sections will provide concrete examples of these constraints. Note that we could have written domains as constraints and even combined all constraints into one, but the above formulation is closer to the mechanics of model solving in CP.

Picturing the model as a (hyper)graph whose vertices are the variables and whose (hyper)edges are the constraints linking the variables in their scope provides insight into the basic algorithm used in CP. Looking locally at a particular edge (constraint), the algorithm attempts to modify the domain of the incident vertices (variables) by removing values which cannot be part of any solution because they would violate that individual constraint; this *local consistency* step can be performed efficiently. The modification of a vertex's domain triggers the inspection of all incident edges, which in turn may modify other domains. This recursive process stops when either all domain modifications have been dealt with or the empty domain is obtained, in which case no solution exists (i.e. the feasible set is empty). The overall behavior is called *constraint propagation*.

Different levels of consistency have been defined to formally describe which values are left in domains – we present the two main ones, after some notation. Let $D^{\min}$ and $D^{\max}$ denote respectively the smallest and largest value in set D. We write $[D] = \{a \in \mathbb{Z} : D^{\min} \le a \le D^{\max}\}$, the smallest discrete interval containing every value in domain D and $[D]^{\mathbb{R}} = \{a \in \mathbb{R} : D^{\min} \le a \le D^{\max}\}$, the smallest real interval containing every value in domain D.

Definition 2.1 (domain consistency)**.** *Constraint* $C(x_1, \ldots, x_m)$ *is* domain consistent *iff for each variable* x_i $_{1 \le i \le m}$ *and for each value* $d_i \in D_i$ *in its domain, there are values* $d_j \in D_j$ *for every other variable* x_j $_{1 \le j \le m, j \ne i}$ *such that* $(d_1, \ldots, d_m) \in C$.

Definition 2.2 (bounds consistency)**.** *Constraint* $C(x_1, \ldots, x_m)$ *is* bounds consistent *iff for each variable* x_i $_{1 \le i \le m}$*, there are values* $d_j \in [D_j]^{\mathbb{R}}$ *for every other variable* x_j $_{1 \le j \le m, j \ne i}$ *such that* $(d_1, \ldots, d_{i-1}, D_i^{\min}, d_{i+1}, \ldots, d_m) \in C$ *and* $(d_1, \ldots, d_{i-1}, D_i^{\max}, d_{i+1}, \ldots, d_m) \in C$.

Note that the "supporting" d_j's in bounds consistency are possibly real values. There is an analogy to be made here with the use of continuous relaxations in IP: whereas domain consistency can be expensive to enforce in general (but see particular cases in Sections 4 and 5) because of its combinatorial nature over a discrete domain, bounds consistency typically only requires simple algebraic manipulations over relaxed continuous domains. However bounds consistency is a strictly weaker property than domain consistency and unless stated otherwise, from now on when we talk of consistency we refer to domain consistency.

Since constraint propagation may stop with indeterminate variables (i.e. whose domain still contains several values), the solution process requires search and its potentially exponential cost. It usually takes the form of a tree search in which branching corresponds to fixing a variable to a value in its domain, thus triggering more constraint propagation. We call *variable-selection heuristic* and *value-selection heuristic* the way one decides which variable to branch on and which value to try first, respectively.

3 Constraint-Centered Branching Heuristics

Until recently, the only visible effect of the consistency algorithms had been on the domains, projecting a constraint's set of solutions on each of the variables. Accordingly, most branching heuristics in CP rely on information at the level of individual variables, essentially looking at the cardinality of their domain or at the number of constraints on them. Constraints play a central role in CP because they encapsulate powerful specialized consistency algorithms but firstly because they bring out the underlying structure of combinatorial problems. That exposed structure could also be exploited during search. Information about the number of solutions to a constraint can help a search heuristic focus on critical parts of a problem or on promising solution fragments. Such *constraint-centered branching heuristics* rely on some knowledge about the number of solutions to individual constraints [22].

In the rest of this article we concentrate on evaluating the number of solutions of certain combinatorial structures (i.e. constraints) for the purpose of constraint-centered branching heuristics. Keep in mind that counting here is not an end in itself but a mean to guide search, and it will be performed repeatedly. Therefore in this context we are willing sometimes to trade some accuracy for speed of execution.

One simple constraint-centered branching heuristic that has been very successful branches first on the variable assignment with the highest solution density overall. But we need the following definitions.

Definition 3.3 (solution count). *Given a constraint $C(x_1,\dots,x_m)$ and respective finite domains D_i $_{1\le i\le m}$, let $\#C$ denote the number of points in $D_1\times\cdots\times D_m$ that are solutions to constraint C, called its* solution count*.*

Definition 3.4 (solution density). *Given a constraint $C(x_1,\dots,x_m)$, respective finite domains D_j $_{1\le j\le m}$, a variable x_i in the scope of C, and a value $d\in D_i$, we will call*

$$\sigma(x_i,d,C)=\frac{\#C(x_1,\dots,x_{i-1},d,x_{i+1},\dots,x_m)}{\#C(x_1,\dots,x_m)}$$

the solution density *of pair (x_i,d) in C. It measures how often a certain assignment is part of a solution.*

As much as possible, we wish these concepts to carry over to polytopes. For the constraints we introduce, corresponding polytopes can be defined with the understanding that each domain D is being relaxed to $[D]$. So strictly speaking we may not be counting over the same structure if a domain has "holes". But with that exception the concept of solution count of a constraint corresponds to counting lattice points in the corresponding polytope. The numerator in the definition of solution density can also be seen as the number of lattice points inside the intersection of the polytope describing C with the $x_i=d$ hyperplane, yielding still another polytope.

Before moving on we mention a few related exceptions to the norm with regards to CP branching heuristics. Kask et al. [5] approximate the solution count of the whole problem given a partial solution and use it in a value selection heuristic, choosing the value whose assignment to the current variable gives the largest approximate solution count. Inspired by the use of pseudo-costs in IP, Refalo [14] proposes a generic variable selection heuristic based on the impact the assignment of a variable has on the reduction of the remaining search space, roughly approximated as the Cartesian product of the domains of the variables.

4 The Frequency-of-Occurrence Polytopes

Problems of a combinatorial nature often make statements about the number of occurrences of certain objects in any solution. This has been recognized in CP and given the status of a few special constraints, the first ones we will investigate.

Definition 4.5 (Global Cardinality Constraint [16]). *The set of feasible tuples of a constraint $\texttt{gcc}(X,D,l,u)$ where $X=\{x_1,\dots,x_m\}$ is a set of variables, $D\subset\mathbb{Z}$ is a set of values, and $l:D\mapsto\mathbb{N}\quad u:D\mapsto\mathbb{N}$ are functions, is defined as:*

$$\begin{aligned}\texttt{gcc}(X,D,l,u)=\{&(d_1,\dots,d_m):d_i\in D_i\ \forall 1\le i\le m,\\ &l(d)\le|\{d_i\ _{1\le i\le m}:d_i=d\}|\le u(d)\ \forall d\in D\}\end{aligned}$$

These functions associate with each value in D a lower and upper bound on the number of times it occurs in a tuple.

Even more frequent is the special case where each value should occur at most once, i.e. the values taken by the given variables should be pairwise different. It is usually considered separately.

Definition 4.6 (All Different Constraint [15]). *The set of feasible tuples of a constraint $\texttt{alldifferent}(X)$ where $X=\{x_1,\dots,x_m\}$ is defined as:*

$$\begin{aligned}&\texttt{alldifferent}(X)\\ &\qquad=\{(d_1,\dots,d_m):d_i\in D_i,\ d_i\neq d_j\ \forall 1\le i\neq j\le m\}\end{aligned}$$

The polytopes of these constraints are well understood and correspond to particular network flow formulations.

Example 4.1. *Consider constraint $\texttt{gcc}(\{x_1,x_2,x_3,x_4\},\{1,2\},l,u)$ with $D_1=D_3=\{1,2\}$, $D_2=D_4=\{1\}$, and $l(1)=1,l(2)=3,u(1)=2,u(2)=5$. There is a natural representation of this as a network with one node for each variable and value, from the source to every variable node an arc with demand and capacity 1, for each value $d\in D_i$ an arc from the "x_i" node to the "d" node with demand 0 and capacity 1, and from every "d" node to the sink an arc with demand $l(d)$ and capacity $u(d)$. Figure 1 illustrates the corresponding network. There is a one-to-one correspondence between solutions to the gcc and feasible flows.*

In the rest of this section we concentrate on counting algorithms for the alldifferent constraint.

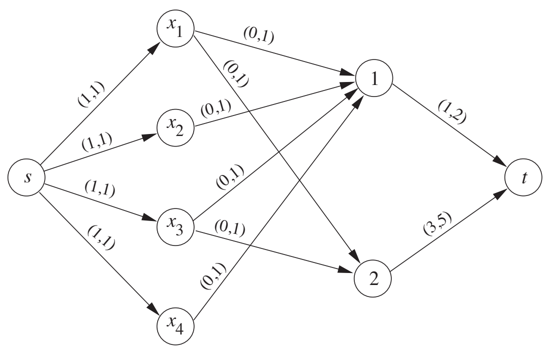

Figure 1. *A feasible network flow problem representation of a* gcc *constraint.*

Definition 4.7 (Value Graph)**.** *Given a set of variables* $X = \{x_1, \dots, x_m\}$ *with respective domains* $D_1, \dots, D_m$*, we define the* value graph *as a bipartite graph* $G = (X \cup D, E)$ *where* $D = \bigcup_{i=1,\dots,m} D_i$ *and* $E = \{\{x_i, d\} : d \in D_i\}$.

There is a natural bijection between a maximum matching of size $|X|$ on the value graph and a solution to the related `alldifferent` constraint, hence algorithms from matching theory are adapted to reach domain consistency and even bounds consistency.

Finding the number of solutions is equivalent to counting the number of maximum matchings on the value graph, which is itself equivalent to the problem of computing the permanent of a square (0-1) matrix A:

$$per(A) = \sum_{j=1}^{m} a_{1,j}\, per(A_{1,j})$$

where $A_{i,j}$ denotes the submatrix obtained from A by removing row i and column j (the permanent of the empty matrix is equal to 1). That problem is $\#P$-hard so we turn to estimates.

4.1 Sampling

Based on the latter recursive definition, Rasmussen proposed in [13] a very simple recursive estimator for the permanent. Despite its simplicity, the estimator is unbiased and in general shows good experimental behavior. Rasmussen also gave some theoretical assurances for dense matrices but still for some specific matrices such as the upper triangular matrix the estimate is likely to be very poor.

A simple way to improve the quality of the approximation is to add propagation to Rasmussen's estimator. After randomly choosing a row i and a column j, we can propagate on the submatrix $A_{i,j}$ in order to remove all the 1-entries (edges) that do not belong to any maximum matching thus achieving domain consistency (the pseudocode is shown in Algorithm 1). This broadens the applicability of the method; in matrices such as the upper triangular matrix, the propagation can easily lead to the identity matrix for which the estimator performs exactly. And because we maintain domain consistency, we reach a feasible sample every time whereas Rasmussen's original algorithm ends up with infeasible solutions whenever it reaches a case in which $W = \emptyset$.

The solution count is then simply

$$\#\texttt{alldifferent} \approx X_A$$

Given the set of solution samples S, we denote by $S_{x_i,d} \subseteq S$ the subset of samples in which $x_i = d$. The solution densities are approximated as:

$$\sigma(x_i, d, \texttt{alldifferent}) \approx \frac{|S_{x_i,d}|}{|S|}$$

The time complexity of the sampling approach is quadratic in the number of variables and linear in the sum of the domain sizes [22].

```
if $m = 0$ then
    $X_A = 1$
else
    Choose $i$ u.a.r. from $\{1 \dots m\}$;
    $W = \{j : a_{i,j} = 1\}$;
    Choose $j$ uniformly at random from $W$;
    Propagate on $A_{i,j}$;
    Compute $X_{A_{i,j}}$;
    $X_A = |W| \cdot X_{A_{i,j}}$;
end
Return $X_A$;
```

Algorithm 1. *Rasmussen's estimator of the solution count of* `alldifferent`*, augmented with propagation*

4.2 Upper Bounds

Even though the sampling approach is fast there is still a computational price to pay compared to simpler branching heuristics, and it can be a disadvantage when solving easy instances. There are known upper bounds on the permanent that are even faster to compute. Minc [7] conjectured and then Brègman [2] proved that the permanent is bounded from above by the following formula:

$$perm(A) \le \prod_{i=1}^{m} (r_i!)^{1/r_i}.$$

where in our case $r_i = |D_i|$. Recently, Liang and Bai [6], inspired by Rasmussen's work, proposed a new upper bound:

$$perm(A)^2 \le \prod_{i=1}^{m} q_i(r_i - q_i + 1).$$

where $q_i = min\{\lceil \frac{r_i+1}{2} \rceil, \lceil \frac{i}{2} \rceil\}$. Since none of the two upper bounds dominates the other, we take their minimum as our alternate approximation of the solution count for an `alldifferent` constraint. By precomputing and storing part of the formulas, each call takes time which is asymptotically linear in the number of variables involved [21].

For their part, solution densities are computed according to their definition but taking the ratio of the two corresponding approximations.

4.3 Counting Accuracy

We provide some empirical evidence for the accuracy of the proposed approaches. A thousand instances over 10 to 20 variables and with domains of various sizes were generated and their lattice points enumerated (some instances had billions of solutions). We measured the relative counting error, the maximum absolute solution density error, and the average absolute solution density error.

Figure 2 shows the counting error for the sampling algorithm and Rasmussen's with varying timeout. The results for the upper bound approach are not reported because they are much worse (from 40 % to 2300 %). Different shades of gray indicate different percentages of removals of values inside the domains; series represent different algorithms and they are grouped based on the varying timeouts. For randomized algorithms we report the average of ten runs.

Figures 3 and 4 show respectively the maximum and average absolute solution density errors. Again the sampling algorithm shows a better accuracy than Rasmussen's. Despite their poor performance in approximating the solution count, upper bounds provide a very good tradeoff between approximation accuracy and computation time when deriving solution densities: giving an equivalent amount of time to the other two, they behave best.

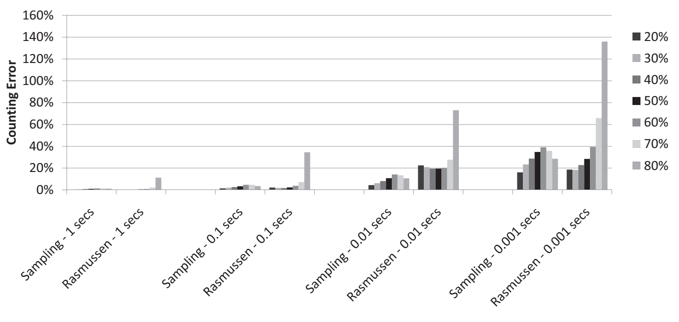

Figure 2. *Relative counting error for one thousand alldifferent instances with varying variable domain sizes.*

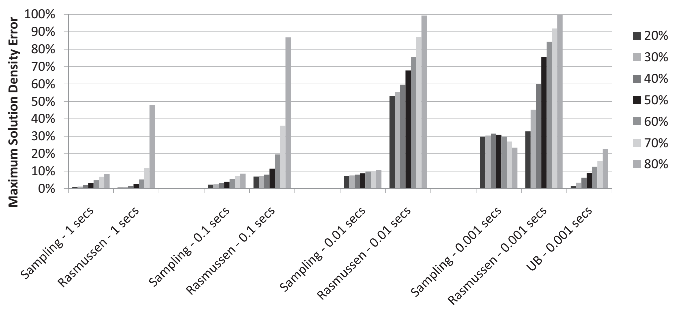

Figure 3. *Maximum absolute solution density error for one thousand alldifferent instances with varying variable domain sizes.*

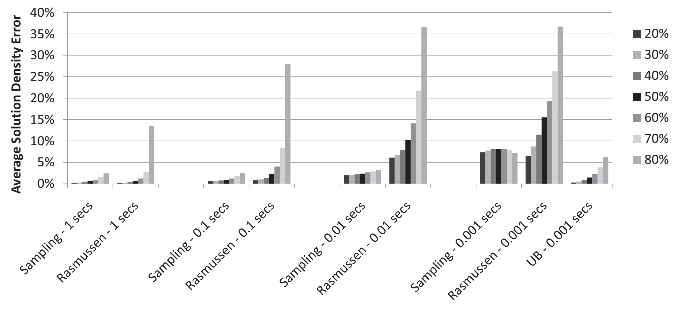

Figure 4. *Average absolute solution density error for one thousand alldifferent instances with varying variable domain sizes.*

5 The Sequencing-Order Polytopes

Some combinatorial problems feature sequencing constraints, restricting the way certain objects are sequenced in a given logical (usually temporal) order. This is frequent in rostering when considering activity types over time; it may also occur in production scheduling e.g. when considering cars with options on an assembly line.

From the realization that a sequence of values can be viewed as a word in some language, researchers in CP turned to the theory of formal languages, which is well-known to computer scientists because of its ties with compiler design and the theory of computation. The result was the introduction of the `regular`[8] and `grammar`[18, 11] constraints, corresponding respectively to regular and context-free languages.[1] We describe the two constraints and discuss counting the lattice points of the first.

Definition 5.8 (Deterministic Finite Automaton). *A deterministic finite automaton is a 5-tuple* $\Pi = (Q, \Sigma, \delta, q_0, F)$ *where* Q *is a finite set of states,* Σ *is an alphabet,* $\delta : Q \times \Sigma \to Q$ *is a partial transition function,* $q_0 \in Q$ *is the initial state, and* $F \subseteq Q$ *is the set of final (or accepting) states. The transition function can be extended to sequences of symbols (words) from the alphabet:* $\delta(q, \langle s_1, s_2, \ldots, s_m \rangle) = \delta(\cdots \delta(\delta(q, s_1), s_2), \ldots, s_m)$. *The (regular) language recognized by* Π *is the set of words* $\{w \in \Sigma^\star : \delta(q_0, w) \in F\}$.

Definition 5.9 (Regular Language Membership Constraint). *The set of feasible tuples for a constraint* `regular`(X, Π) *where* $\mathbf{x} = (x_1, \ldots, x_m)$ *and* Π *is a finite automaton is defined as:*

$$\texttt{regular}(\mathbf{x}, \Pi) = \{(d_1, \ldots, d_m) : d_i \in D_i,\ d_1 d_2 \cdots d_m \textit{ belongs to the language recognized by } \Pi\}$$

The consistency algorithm associated to this constraint is based on the computation of paths in a graph. The automaton is unfolded into a layered acyclic directed graph $G = (V, A)$ where vertices of a layer correspond to states of the automaton and an arc joining a vertex of layer L_i to another of layer L_{i+1} represents a feasible value for variable x_i. We denote by $v_{\ell,q}$ the vertex corresponding to state q in layer ℓ. The first layer only contains one vertex, v_{1,q_0}; the last layer only contains vertices corresponding to accepting states, $v_{m+1,q}$ with $q \in F$. This graph has the property that paths from the first layer to the last are in one-to-one correspondence with solutions of the constraint. (And any arc not belonging to such a path is removed through a forward/backward sweep of the graph.) Figure 5 gives a very small example of a layered directed graph built for one such constraint on five variables.

As for its polytope, a network flow formulation is given in [3] for the `regular` constraint, following closely the structure of the graph described above.

Definition 5.10 (Context-Free Grammar). *A context-free grammar is a 4-tuple* $\mathcal{G} = (\Sigma, N, S, P)$ *where* Σ *is an alphabet (the set of terminal symbols),* N *is a set of non-terminal symbols,* $S \in N$ *is the starting symbol, and* P *is a set of productions of the form* $A \to w$ *where* $A \in N$ *is a non-terminal symbol and* w *is a sequence of terminal and non-terminal symbols. The (context-free) language recognized by G is the set of words reproduced by the leaves of parsing trees for G (obtained through the application of productions to non-terminal symbols, starting from* S*).*

Definition 5.11 (Context-Free Language Membership Constraint). *The set of feasible tuples for a constraint* `grammar`$(\mathbf{x}, \mathcal{G})$ *where* $\mathbf{x} = (x_1, \ldots, x_m)$ *and G is a context-free grammar is defined as:*

$$\texttt{grammar}(\mathbf{x}, \mathcal{G}) = \{(d_1, \ldots, d_m) : d_i \in D_i,\ d_1 d_2 \cdots d_m \textit{ belongs to the language recognized by } \mathcal{G}\}$$

The consistency algorithm for the `grammar` constraint is based on the CYK parser and builds an and/or graph that encodes every possible parsing tree [12]. That graph was linearized in [3] in order to obtain the polytope of the `grammar` constraint.

5.1 Counting Lattice Points for `regular`

Given the graph built by the consistency algorithm for `regular`, what is the additional computational cost of determining its number of solutions? As we already pointed out, every (complete) path in that graph corresponds to a solution. Therefore it is sufficient to count the number of such paths. We express this through a simple recurrence relation, which we can compute by dynamic programming. Let $\#op(\ell, q)$ denote the number of paths from $v_{\ell,q}$ to a vertex in the last layer. Then we have:

$$\#op(m+1, q) = 1$$
$$\#op(\ell, q) = \sum_{(v_{\ell,q}, v_{\ell+1,q'}) \in A} \#op(\ell+1, q'), \quad 1 \le \ell \le m$$

We compute $\#ip(\ell, q)$, the number of paths from v_{1,q_0} to $v_{\ell,q}$, in a similar way.

The solution count is given by

$$\#\texttt{regular} = \#op(1, q_0)$$

in time linear in the size of the graph even though there may be exponentially many of them. Therefore this is absorbed in the asymptotic complexity of the consistency algorithm.

In the graph of `regular`, variable-value pair (x_i, d) is represented by the arcs between layers i and $i+1$ corresponding to transitions on value d. The number of solutions in which $x_i = d$ is thus equal to the number of paths going through one of those arcs. Consider one such arc $(v_{i,q}, v_{i+1,q'})$: the number of paths through it is the product of the number of outgoing paths from $v_{i+1,q'}$ and the number of incoming paths to $v_{i,q}$.

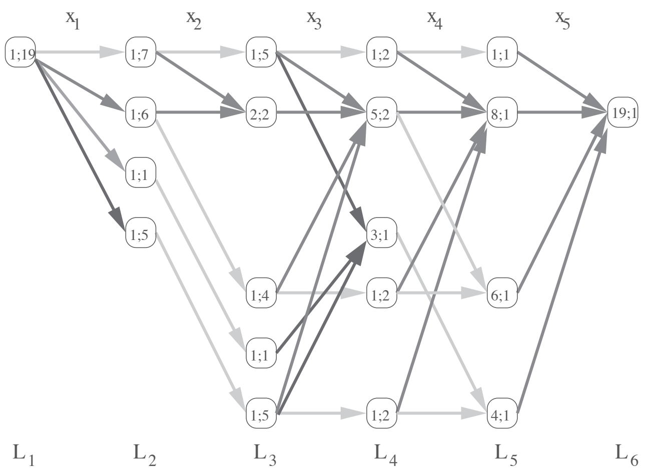

Figure 5. *Graph of a* `regular` *constraint*

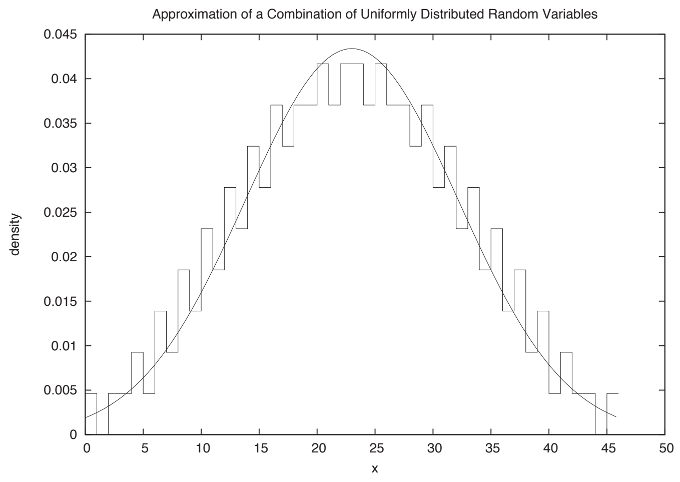

Figure 6. *The histogram is the actual distribution of the expression* $3x + 4y + 2z$ *for* $x, y, z \in [0, 5]$. *The curve is the approximation given by the Gaussian curve with mean* $\mu = 22.5$ *and variance* $\sigma^2 = 84.583$.

Let $A(i, d) \subset A$ denote the set of arcs representing variable-value pair (x_i, d). The solution density of pair (x_i, d) is thus given by

$$\sigma(x_i, d, \texttt{regular}) = \frac{\sum_{(v_{i,q}, v_{i+1,q'}) \in A(i,d)} \#ip(i, q) \cdot \#op(i + 1, q')}{\#op(1, q_0)}$$

Once the $\#ip();\#op()$ values are tabulated (they are shown as vertex labels in Figure 5), the time to compute the solution density of a given pair is in the worst case linear in $|Q|$, the number of states of the automaton [22].

6 The Knapsack Polytopes

Knapsack constraints are very common in IP formulations of combinatorial problems and they also occur in CP. Here the polytope is immediate.

Definition 6.12 (knapsack constraint). *The set of feasible tuples for a constraint* knapsack$(\mathbf{x}, \mathbf{c}, \ell, u)$, *where* $\mathbf{x}$ *is a column vector of finite domain variables* $(x_1, x_2, \ldots, x_m)^T$, $\mathbf{c} = (c_1, c_2, \ldots, c_m)$ *is an integer row vector, and* ℓ *and* u *are integers*[2], *is defined as:*

$$\text{knapsack}(\mathbf{x}, \mathbf{c}, \ell, u) = \{(d_1, \ldots, d_m) \ : \ d_i \in D_i, \ \ell \le \sum_{i=1}^{m} c_i d_i \le u\}$$

Strictly speaking the coefficients c_i and the variables x_i are non-negative but the algorithms we describe in this section also work if that restriction is lifted.

6.1 Counting Lattice Points for knapsack

In [19], Trick proposes an algorithm to achieve domain consistency for knapsack constraints that relies on a graph whose structure is very similar to that of the `regular` constraint. Not surprisingly then, the computation of exact solution counts and solution densities for knapsack constraints follows quite directly from Section 5.1. However in this case the algorithm runs in pseudopolynomial time since the size of the graph also depends on the magnitude of the knapsack bounds (and of the coefficients and domain values if we allow them to be negative). We refer the reader to [9] for details.

6.2 Estimating Lattice Points for knapsack

But knapsack constraints in CP have traditionally been handled by enforcing bounds consistency, much cheaper than domain consistency when the domains of the variables are large. So if the graph mentioned above is not built, how do we count solutions?

We first relax each domain D to $[D]$, the smallest discrete interval containing it. Consider variable x whose domain is the discrete interval $[a, b]$ and assume that each domain value is equiprobable. We associate to x the discrete random variable X which follows a discrete uniform distribution with probability mass function

$$f(v) = \begin{cases} \frac{1}{b-a+1} & \text{if } a \le v \le b \\ 0 & \text{otherwise} \end{cases},$$

mean $\mu = \frac{a+b}{2}$, and variance $\sigma^2 = \frac{(b-a+1)^2-1}{12}$.

To find the distribution of a variable subject to a knapsack constraint, one needs to find the distribution of a linear combination of uniformly distributed random variables. Lyapunov's Central Limit Theorem allows us to approximate the distribution of such a linear combination.

Theorem 6.1 (Lyapunov's Central Limit Theorem). *Consider the independent random variables* $X_1, \ldots, X_n$. *Let* μ_i *be the mean of* X_i, σ_i^2 *be its variance, and* $r_i^3 = E[|X_i - \mu_i|^3]$ *be its third central moment. If*

$$\lim_{n\to\infty} \frac{(\sum_{i=1}^n r_i^3)^{\frac{1}{3}}}{(\sum_{i=1}^n \sigma_i^2)^{\frac{1}{2}}} = 0$$

then the random variable $S = \sum_{i=1}^n X_i$ *follows a normal distribution with mean* $\mu_S = \sum_{i=1}^n \mu_i$ *and variance* $\sigma_S^2 = \sum_{i=1}^n \sigma_i^2$.

Note that this theorem does not assume that the variables are taken from identical distributions. This is necessary since variables with different domains have different distributions. Let $Y = \sum_{i=1}^n c_i X_i$ be a random variable where X_i is a discrete random variable uniformly chosen from the interval $[a_i, b_i]$ and c_i is a non-negative coefficient. When n tends to infinity, it can be shown that the distribution of Y tends to a normal distribution with mean $\sum_{i=1}^n c_i \frac{a_i+b_i}{2}$ and variance $\sum_{i=1}^n c_i^2 \frac{(b_i-a_i+1)^2-1}{12}$.

Consider the knapsack constraint $\ell \le \sum_{i=1}^m c_i x_i \le u$. Introduce variable x_{m+1} with domain $[\ell, u]$. We obtain $x_j = \frac{1}{c_j}(x_{m+1} - \sum_{i=1}^{j-1} c_i x_i - \sum_{i=j+1}^m c_i x_i)$. Some coefficients in this expression might be negative. They can be made positive by using $c_i' = -c_i$ and $[D_i]' = [-D_i^{\max}, -D_i^{\min}]$. When k grows to infinity, the distribution of x_j tends to a normal distribution as stated before. In practice the normal distribution is a good estimation even for small values of k. For example Figure 6 shows the actual distribution of the expression $3x + 4y + 2z$ for $x, y, z \in [0, 5]$ and its approximation by a normal distribution.

With this in hand, we cannot quite count solutions but we can quickly identify an assignment of highest solution density (for relaxed domains) to branch on which, after all, is our motivation here. The resulting algorithm is linear in the number of variables and logarithmic in the size of the domains [9].

Interestingly, this work on knapsack constraints recently inspired new branching direction heuristics for mixed integer programs [10].

There is much work left to design counting algorithms for other frequently occurring polytopes in CP models and to find the best way to exploit that information in order to guide search. Early results suggest that such an approach may significantly advance the state of the art in CP search.

Notes

1. However we are restricting the language to finite words of some given length.
2. We assume that l and u are finite as they can always be set to the smallest and largest value that cx can take.

Gilles Pesant, Department of Computer and Software Engineering, École Polytechnique de Montréal, Canada. gilles.pesant@polymtl.ca

References

[1] F. Benhamou (ed.), *Principles and Practice of Constraint Programming - CP 2006, 12th International Conference, CP 2006, Nantes, France, September 25–29, 2006, Proceedings*, Lecture Notes in Computer Science, vol. 4204, Springer, 2006.

[2] L.M. Brègman, *Some Properties of Nonnegative Matrices and their Permanents*, Soviet Mathematics Doklady **14** (1973), no. 4, 945–949.

[3] M.-C. Côté, B. Gendron, C.-G. Quimper, and L.-M. Rousseau, *Formal Languages for Integer Programming Modeling of Shift Scheduling Problems*, Constraints (forthcoming).

[4] J. De Loera, *Counting and Estimating Lattice Points: Algebraic and Probabilistic Tools*, Optima (2009).

[5] K. Kask, R. Dechter, and V. Gogate, *Counting-Based Look-Ahead Schemes for Constraint Satisfaction*, in Wallace [20], pp. 317–331.

[6] H. Liang and F. Bai, *An Upper Bound for the Permanent of (0,1)-Matrices*, Linear Algebra and its Applications **377** (2004), 291–295.

[7] H. Minc, *Upper Bounds for Permanents of (0, 1)-matrices*, Bulletin of the American Mathematical Society **69** (1963), 789–791.

[8] G. Pesant, *A Regular Language Membership Constraint for Finite Sequences of Variables*, in Wallace [20], pp. 482–495.

[9] G. Pesant and C.-G. Quimper, *Counting Solutions of Knapsack Constraints*, CPAIOR (L. Perron and M. A. Trick, eds.), Lecture Notes in Computer Science, vol. 5015, Springer, 2008, pp. 203–217.

[10] J. Pryor, *Branching Variable Direction Selection in Mixed Integer Programming*, Master's thesis, Carleton University, 2009.

[11] C.-G. Quimper and T. Walsh, *Global Grammar Constraints*, in Benhamou [1], pp. 751–755.

[12] ———, *Decomposing Global Grammar Constraints*, CP (C. Bessière, ed.), Lecture Notes in Computer Science, vol. 4741, Springer, 2007, pp. 590–604.

[13] L. E. Rasmussen, *Approximating the Permanent: A Simple Approach* , Random Structures and Algorithms **5** (1994), no. 2, 349–361.

[14] P. Refalo, *Impact-Based Search Strategies for Constraint Programming*, in Wallace [20], pp. 557–571.

[15] J.-C. Régin, *A Filtering Algorithm for Constraints of Difference in CSPs*, AAAI, 1994, pp. 362–367.

[16] ———, *Generalized Arc Consistency for Global Cardinality Constraint*, AAAI/IAAI, Vol. 1, 1996, pp. 209–215.

[17] F. Rossi, P. van Beek, and T. Walsh (eds.), *Handbook of Constraint Programming* , Elsevier, 2006.

[18] M. Sellmann, *The Theory of Grammar Constraints*, in Benhamou [1], pp. 530–544.

[19] M.A. Trick, *A Dynamic Programming Approach for Consistency and Propagation for Knapsack Constraints*, Annals of Operations Research **118** (2003), 73–84.

[20] M. Wallace (ed.), *Principles and Practice of Constraint Programming - CP 2004, 10th International Conference, CP 2004, Toronto, Canada, September 27 - October 1, 2004, Proceedings*, Lecture Notes in Computer Science, vol. 3258, Springer, 2004.

[21] A. Zanarini, *Exploiting Global Constraints for Search and Propagation*, Ph.D. thesis, École Polytechnique de Montréal, 2010.

[22] A. Zanarini and G. Pesant, *Solution counting algorithms for constraint-centered search heuristics*, Constraints **14** (2009), no. 3, 392–413.

Main Conference Themes

- Design Optimization
- Identification and Inverse Problems
- Numerical Optimization Techniques
- Efficient Analysis and Reanalysis Techniques
- Sensitivity Analysis
- Industrial Applications

Venue

Congress Centre, Instituto Superior Técnico, Lisbon, Portugal.

Secretariat

Centre for Mechanical Design
Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisbon, Portugal
Phone: +351 218417914; Fax: +351 218417915
Email: engopt2010@dem.ist.utl.pt

INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

About EngOpt 2010

The main goal of EngOpt conferences is to periodically bring together engineers, applied mathematicians and computer scientists working on research, development and practical application of optimization methods applied to all engineering disciplines or developing basic techniques in this field.

Contributions on Engineering Design Optimization, MDO – Multidisciplinary Design Optimization, Inverse Problems, Engineering Simulation Involving Optimization Techniques and Basic Numerical Techniques in all disciplines of engineering are invited. Practical applications are strongly encouraged.

Abstract Submission Deadline:	March 5, 2010
Notification of Acceptance:	April 2, 2010
Full Paper Submission Deadline:	June 4, 2010

Conference Chairmen

Hélder Rodrigues, Instituto Superior Técnico, Portugal
José Herskovits, COPPE - Federal University of Rio de Janeiro, Brazil
Cristóvão M. Mota Soares, Instituto Superior Técnico, Portugal

For complete and up-to-date information, please visit the conference website:

www.engopt2010.org

MOPTA 2010

Bethlehem PA, 18-20 August, 2010.

LEHIGH UNIVERSITY.

MOPTA

The MOPTA conference (**M**odeling and **Opt**imization: **T**heory and **A**pplications) aims to bring together people from both discrete and continuous optimization, working on both theoretical and applied aspects.

Plenary speakers

Egon Balas (Carnegie Mellon)
Mung Chiang (Princeton)
Donald Goldfarb (Columbia U.)
Arkadi Nemirovski (Georgia Tech)
Anthony T. Patera (MIT)
H. Edwin Romeijn (U. of Michigan)
Andrzej Ruszczynski (Rutgers U.)

Modeling competition

Join the student modeling competition at MOPTA! Your team will receive an AIMMS license and have the chance of modeling a real-world optimization problem.

The finalist teams will present their work at MOPTA, where the prize for the best work will be awarded.

Venue

Bethlehem is in the Lehigh Valley, Pennsylvania, and is conveniently located an hour's drive from NYC and Philadelphia. It is easily reached from Allentown (ABE), JFK, Newark, La Guardia, and Philadelphia airports.

Practical information

Abstract submission:	June 16, 2010
Early registration:	July 15, 2010
Conference:	Aug. 18-20, 2010

Contact:

ISE Dept., Lehigh University
200 Packer Ave, Bethlehem PA 18015
Email: `mopta@lehigh.edu`
Phone: 610-758-3865

Organizing committee

Tamás Terlaky (chair)
Pietro Belotti
Frank E. Curtis
Imre Pólik
Ted Ralphs
Larry Snyder
Robert H. Storer
Aurélie Thiele

`http://mopta.ie.lehigh.edu`

Application for Membership

I wish to enroll as a member of the Society. My subscription is for my personal use and not for the benefit of any library or institution.

☐ I will pay my membership dues on receipt of your invoice.
☐ I wish to pay by credit card (Master/Euro or Visa).

Credit card no. *Expiration date*

Family name

Mailing address

Telephone no. *Telefax no.*

E-mail

Signature

Mail to:
Mathematical Programming Society
3600 University City Sciences Center
Philadelphia, PA 19104-2688
USA

Cheques or money orders should be made payable to The Mathematical Programming Society, Inc. Dues for 2009, including subscription to the journal *Mathematical Programming*, are US $ 90. Retired are $ 45. Student applications: Dues are $ 22.50. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status

Institution

A Special Workshop in Honor of Paul Tseng

Large Scale Optimization: Analysis, Algorithms and Applications

Fudan University, Shanghai, China
May 21, 2010

This is a special workshop dedicated to Professor Paul Tseng, University of Washington, who went missing while on a kayak trip in Jinsha river, China, on August 13, 2009. The workshop will be held on May 21, 2010, in Fudan University, Shanghai, China, right before the 2010 Symposium of Chinese Mathematical Programming Society, to be held in Shanghai University, May 22–25, 2010. The workshop will be a forum for Paul's friends and colleagues to share reminiscences and pay tribute to an extraordinary individual and an outstanding researcher.

Professor Paul Tseng has made many fundamental contributions to the theory and algorithms for large scale optimization. This special workshop will feature state of the art research in several major topic areas of large scale optimization where Paul Tseng's work has had a major impact. These include

(1) efficient algorithms for structured convex programs and network flow problems,
(2) error bounds and convergence analysis of iterative algorithms for optimization problems and variational inequalities,
(3) interior point methods and semidefinite relaxations for hard quadratic and matrix optimization problems, and
(4) the applications of large scale optimization techniques in signal processing and machine learning.

The workshop will consist of presentations by invited speakers, and of a poster session that will address these topics. We invite papers to the poster session, especially from friends, students and colleagues of Paul Tseng (please send a message regarding a proposed poster presentation to the organizers). We expect to select a subset of the research articles presented in this workshop for a forthcoming issue of *Mathematical Programming*, Series B, (co-guest editors: Zhi-Quan Luo and Dimitri Bertsekas), and a special issue of *Pacific Journal of Optimization* (co-guest editors: Shuzhong Zhang and Zhi-Quan Luo).

Confirmed invited speakers include:

Dimitri Bertsekas (MIT)
Xin Chen (UIUC)
Duan Li (Chinese University of Hong Kong)
Jong-Shi Pang (UIUC)
Terry Rockafellar (Univ of Washington)
Steve Wright (Univ of Wisconsin)
Ya-Xiang Yuan (Chinese Academy of Sciences)
Jein-Shan Chen (National Taiwan Normal Univ)
Masao Fukushima (Kyoto Univ)
Angelia Nedic (UIUC)
Liqun Qi (Hong Kong Poly Univ)
Jie Sun (National Univ Singapore)
Yinyu Ye (Stanford Univ)
Yin Zhang (Rice Univ)

Attendance to the workshop is free. We have limited funds to partially cover the local expenses of workshop participants. Any inquiries should be sent to the workshop organizers:

Zhi-Quan (Tom) Luo, University of Minnesota (luozq@umn.edu)
Shuzhong Zhang, CUHK (zhang@se.cuhk.edu.hk)

IMPRINT
Editor: Andrea Lodi, DEIS University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy. andrea.lodi@unibo.it
Co-Editors: Alberto Caprara, DEIS University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy. alberto.caprara@unibo.it
Katya Scheinberg, Department of Industrial Engineering and Operations Research, Columbia University, 500 W 120th Street, New York, NY, 10027. katyascheinberg@gmail.com
Founding Editor: Donald W. Hearn

Published by the Mathematical Programming Society.
Design and typesetting by Christoph Eyrich, Mehringdamm 57 / Hof 3, 10961 Berlin, Germany. optima@0x45.de
Printed by Oktoberdruck AG, Berlin.