SIAG/OPT Views-and-News

A Forum for the SIAM Activity Group on Optimization

Volume 16 Numbers 1-2 October 2005

Contents

Tributes to George Dantzig and Leonid Khachiyan

George Dantzig: A Personal Perspective

Walter Murray
Department of Management Science and Engineering, Terman Center, Stanford University, Stanford, CA 94305, USA (walter@stanford.edu).

My earliest memory of George is from 1969. He was being helped by Dick Cottle (a role Dick played the whole time I knew George) into a boat going to the Island of Bender, which is just off Bandol in the south of France. Even then he seemed a frail old man. It was only later I came to realize that George was a lot tougher both physically and mentally than his outward appearance would suggest.

It is not my intent here to catalog George's mathematical contribution. Anyone interested in that could not do better than to start with Dick's recent book "The Basic George B. Dantzig". I shall attempt to shine some light on one facet, the view from where I stood. I would not be writing this if George was simply a great mathematician, to me he was much more and it is this extra dimension that set him apart. Nonetheless, I shall make one mathematical comment. To me George had impeccable mathematical taste and instinct. This is not entirely divorced from his character. He was patient and always took the long view. Perhaps he knew he would live to be ninety. This contrasts sharply with the almost frenetic rush to publish quickly that now permeates much of science. George was driven by curiosity and had much in common with his namesake “Curious George”.

Even when George was alive the celebrations of his many birthday milestones gave occasion to reflect on George’s life. He has been repeatedly referred to as the “Father of Linear Programming”. For his 90th birthday, I commented that perhaps more correctly George should be referred to as the “Father of Linear Programmers”. In all the years I knew George I never heard him make one derogatory remark about anyone in the field (and there were many occasions when he would have had just cause). It was as if we were part of his family and George never spoke ill of his family. George was an intensely loyal man. It is not that George was incapable of being derogatory; indeed his invective on some topics such as lawyers or the higher administration of the University was, in typical George fashion, at an extreme point. Of course it was always delivered in a calm quiet voice. George never needed to raise his voice to get attention. He was in particularly good form when Condoleezza Rice was Provost and she forced the OR Department to merge with the EES Department. George never used his status to get his own way and was always happy to rely only on the force of his argument.

I now think that “Father” is not quite the correct term. Fathers are often strict and need to keep their children on the path they have determined. In my own life George reminds me much more of my Grandmother. She had lived though two world wars and outlived two husbands (my two Grandfathers). Nothing I ever did fazed her in the least. Whenever I was in trouble, which was quite frequent, I always went to her first. She would always be on my side no matter how wrong I had been. She had her idiosyncrasies, just like George, but that just made her all the more endearing.

George was fun. He always had a half smile on his face as if he was mentally recalling some amusing story. You could always tell he was about to say something outrageous when he pursed his lips to try and suppress the smile. Faculty meetings with George present were never boring. George liked to take a rise out of certain people. He only chose those he knew would take it in the spirit it was intended and could handle the public embarrassment it sometimes entailed. His favorite targets were Pete Veinott and Curtis Eaves. Both laughed louder than the rest of us when the comments were made. Of course neither could respond in kind. That would have been like punching a teddy bear.

George’s concern for people in the field seemed to have no limits. It was not just lowly people, but would be extended to everyone, including senior faculty. If he thought somebody’s work was suffering or they were not getting sufficient grant money, then George would find some way to help, often without them knowing. It was not just his willingness to help but the fact he must have been continually checking on everybody to see they were all right. When George asked how you were doing he really wanted to know. Had George not been Jewish he would have got my vote for Pope.

George wrote wonderful letters for people, even for people he did not know. If I ever wanted support for some cause, such as a student applying for a fellowship, I could always rely on George to put his weight behind it. Given that he was very busy he sometimes would ask me to write the letter and would sign anything I wrote without question. It was always easier and more fun to write a letter in George’s name than my own. I just had to lay it on with a trowel. Maybe George knew Disraeli.

His attitude to the efforts of people in the field was like that of a doting parent, whatever the quality of the work he always saw the positive side of it. Michael Saunders and I once pointed out to George that the work of a student was really very similar to some work that had been published. George’s response was to praise the student for rediscovering not any old rubbish but really good stuff. It may be that the discovery of the Simplex method was due to George’s positive outlook. George told me that Von Neumann had commented that had he discovered the Simplex algorithm he would have dismissed it as impractical.

It was not just students who George encouraged. Over the years George must have received a lot of mail and met many people from many walks of life who suggested improvements in the Simplex algorithm and other things. He listened to them patiently much as you would with a child. As recently as 1999 I got email from a professor at a foreign university who claimed to have discovered how to modify the Simplex algorithm to avoid artificial variables and that the new algorithm did not appear to need more iterations than the number of rows or variables. The evidence supporting this claim came from tests on problems with no more than ten rows or variables. The message concluded by mentioning he had spoken recently with Prof. Dantzig about the method and George had found it interesting and suggested he contact me. This was typical of George being a little mischievous.

George slowed physically towards the end but his mind and wit remained as sharp as ever. He also refused to let his physical infirmity hinder him. The extensive celebrations for 90th birthday that he attended were a testimony to that. A few months earlier he had also attended the 50th birthday party of Mukund Thapa. Mukund's parties always have a loud rock band, which George hated, and Indian food, which was also something not to George's taste. Nonetheless George showed up. Each time he attended Mukund's parties he would pointedly tear off the corners of a paper serviette, form them into plugs and then place them in his ears. Remarkably he could still carry on a conversation.

In the last year or so George had trouble walking without assistance. When he needed help, someone would put their arm around him and he would put his arm on their shoulder. Then while hugging him you would shuffle along. Even if the distance was short the journey could take some time. He did not seem to resent the need to be helped and it was an opportunity to demonstrate affection for George. The last time I helped George was when we were at Dick's house to celebrate George's 90th birthday. We were in the garden and George needed to move from there through the kitchen, the hall and into the dining room. George would use these opportunities to chat. It always seemed very personal since being physically very close we talked in a whisper. Whatever the conversation, it always ended with the same phase, "Whatever you do Walter, do not grow old". I am not sure what alternatives George thought I had, but I did not like to disagree with him. If he had seen me skiing or driving, he would have seen I was doing my best to comply. I did wonder if he thought it was just me unsuitable for old age or that he thought it good advice for everyone. Maybe he was a fan of the film "Logan's Run" or subscribed to the idea it would be better if we were all born old and got progressively younger

The last time I saw George was when he was in the hospital just prior to his death. I was accompanied by Peter Glynn and Gerd Infanger. George was attached to a number of tubes and was obviously heavily medicated. He was drifting in and out of consciousness and was struggling to breath. His frail body barely made a ripple in the blanket that covered him. It was a hard sight to observe. After some time the Nurse said "Prof. Dantzig, Peter, Gerd and Walter are here to see you.". There was a brief pause and then in a clear voice George said, "I am overwhelmed.". Even in his distressed state George was trying to make us feel better. A few days later George died at home. His last words were to ask his caregiver if it was all right to leave now. Having been told that it was, he then added, "Will you miss me?". Courage was described by Hemingway as grace under pressure. There are few who have the grace of George.

I never heard George complain (and you do not get to ninety without having a lot to moan about), raise his voice, be in a bad temper, or not be pleased to see me. He was a fabulous human being and if I had to choose between inheriting his mathematical talent or his human qualities I would not hesitate in choosing the latter.

Leonid Khachiyan, 1952–2005: An Appreciation

Michael Todd
School of Operations Research, Cornell University, Ithaca, NY 14953, USA (miketodd@cs.cornell.edu).

Leonid Khachiyan died of a heart attack a few days before his 53rd birthday in South Brunswick, NJ. He is survived by his wife of 20 years, Olga Pischikova Reynberg, and teenage daughters Anna and Nina, student and student-to-be at Rutgers University, where Khachiyan had taught since 1990. Previously he was a researcher at the Computing Center of the USSR Academy of Sciences, an adjunct professor at the Moscow Institute of Physics and Technology, and a visiting scientist at Cornell University.

This article is a tribute to Leo Khachiyan as a friend and an optimizer. I'll also give references for some of his key papers.

Leo was famous in the optimization community for his use of the ellipsoid algorithm to demonstrate that linear programming, in the Turing machine model, had a polynomial-time algorithm; for this work, he received the Fulkerson Prize of the American Mathematical Society and the Mathematical Programming Society. This was an astonishing result, not only in settling a long-open problem in complexity, but also in introducing radically new viewpoints and techniques to linear programming. While the ellipsoid method had been developed by David Yudin and Arkadi Nemirovski and, independently, by Naum Shor in 1976–77, for convex optimization, Khachiyan used it in a tour-de-force to crack the complexity problem for linear programming. Since the algorithm was designed for the real-number model, and required an estimate of the distance to an optimal solution, Khachiyan had to establish a number of bounds on sizes of solutions, volumes of polyhedra, and the precision required to carry out the computations, to achieve his goal. The result was first published in a 4-page note without proofs in Soviet Mathematics Doklady in February 1979 [3]. It was brought to the attention of Western researchers in a presentation at the Montreal Mathematical Programming Symposium in August 1979 and in a later publication by Peter Gács and Laci Lovász. Their presentation was far clearer than the original to those not used to thinking in the varying coordinate systems viewpoint of the Soviet researchers. Khachiyan's later 1980 paper [4] in the journal USSR Computational Mathematics and Mathematical Physics provided the proofs for the results in his earlier work.

After its development in 1947 by George Dantzig, the simplex method had sloughed off the challenge of a number of alternative algorithms, notably iterative methods based on fictitious play in 2-person games, in the '50s, and had found itself successfully applied to a wider range of vastly larger-scale problems through the '50s and '60s. Then, in the '70s, it ran into a theoretical no-man's-land with the newfound notion of polynomial-time algorithms and Victor Klee and George Minty's discovery of a class of problems for which Dantzig's pivot rule for the simplex method led to an exponential number of pivots. While more recent versions, such as the dual steepest-edge variant that appears to be the best at present, remain highly competitive with the more recent interior-point methods and an indispensable part of the arsenal of any optimizer, they still exhibit exponential behavior on some examples. (To some extent, their good behavior in practice has been explained via analyses of the expected behavior of the simplex method by a number of authors, and by the more recent smoothed analysis of Daniel Spielman and Shang-Hua Teng.)

Leo's result was a bombshell in this environment. The use of the ellipsoid method, with its approximation of the polyhedral feasible region by ellipsoids, seemed counter to all we held dear: vertices, edges, phase 1 – phase 2, and even finite convergence to an exact solution in exact arithmetic. Instead we had to start with gigantic spheres, and then generate a sequence of shrinking ellipsoids until one was found sufficiently small that its center could be rounded to give an exact solution — assuming that all the data was rational. This was a pretty wild way to approach a problem that we knew had a finite solution via pivoting, and in fact bore some resemblance to the iterative methods tried in the '50s, but with a twist: the changing shapes of the ellipsoids gave a sort of variable-metric slant to the earlier relaxation methods.

It was natural that such a result would get a huge amount of press. Linear programming was big business, and leading papers around the world tried to educate their readers to the significance of the result, with very spotty results. The ensuing brouhaha has been well documented in Gene Lawler's article [7]. The effect on the optimization community was more rational. Many people tried, and failed, to turn Leo's result into a practical method for the solution of large-scale linear programming problems. (Part of the problem lies in the fact that the algorithm seems to require in practice a number of iterations close to its worst case bound: it also leads to very ill-conditioned linear systems.) A lot of attention was turned to the amazing theory of Yudin and Nemirovski on the informational complexity of nonlinear programming. And a few people, notably Martin Grötschel, Laci Lovász, and Lex Schrijver, realized that the ellipsoid method could be used as a powerful tool in combinatorial optimization, thereby lending a (very) little credence to some of the outrageous claims that had been made in the popular press. (Just one example: the Guardian headlined its story: "Soviet Answer to the 'Traveling Salesmen.' " Of course, the ellipsoid method has not shed any light on the complexity of the Traveling Salesman Problem.) And the ellipsoid method was the first theoretically good algorithm for the burgeoning field of semidefinite programming.

So Khachiyan became famous: but what of his other research and its significance? Interestingly, his first work was concerned with the convergence rate of iterative processes for solving matrix games, and he obtained some negative results: the error decreased at best inversely with the iteration count. His fourth paper, at the age of 26, was the Doklady announcement that LP was in P. The ideas in that work, estimating the sizes of solutions, looking at rational or integer solutions, and using geometric ideas in combinatorics and optimization, appear in much of Leo's subsequent research. He extended the polynomial algorithm to convex quadratic programming with M. K. Kozlov and S. P. Tarasov, and then considered the size of solutions and the complexity of solving convex polynomial programming problems, in either continuous or integer variables. He wrote a lovely survey of results in this area for the Proceedings of the 1983 International Congress of Mathematicians [5]. Let me mention a couple of results from that work. He bounded the size of a solution to a system of convex polynomial inequalities by a 2-stage exponential function, and showed by a simple example ($x_1 \geq h$, $x_2 \geq x_1^2$, ..., $x_n \geq x_{n-1}^2$) that this was the best possible. Yet he showed that such a solution when the degree was fixed could be "compactly represented" in polynomial space using a solution of just polynomial size to a reduced subsystem, consistent if and only if the original system was; moreover, this characterization could be found by a polynomial algorithm. Finally, he extended Lenstra's well-known result in integer programming by showing that there was a polynomial algorithm for finding an exact solution to a convex polynomial programming problem in real and/or integer variables, if the degree and the number of variables was fixed.

Another beautiful result [9], with Tarasov and I. I. Erlikh, replaced the sequence of circumscribing ellipsoids in the usual ellipsoid method with a sequence of inscribed ellipsoids (each inscribed in the current localizing set). This method allowed a decrease in the complexity of approximately solving a convex minimization problem by the factor n, the dimension of the problem, and thereby obtained the optimal (worst-case) complexity. The cost was that each iteration required the finding of an (approximately) largest volume inscribed ellipsoid; the authors suggested doing this via the original ellipsoid method (but without further function oracle calls)! This paper also had a surprising geometric theorem concerning volumes of inscribed ellipsoids, which Leo later improved. This concern with volumes led to later work on the complexity of polytope volume computation and on the conductance of Markov chains (involving another neat geometric inequality) to bound the mixing time for randomized methods.

After coming to the States, Leo's work continued some of its old themes, like his work on the complexity of maximal volume ellipsoids inscribed in a polytope and his fascinating paper on rounding polytopes [6], and added some new ones. He wrote a series of papers with Bahman Kalantari on various matrix scaling and balancing problems, and a series of papers with Mike Grigoriadis on fast approximations of multicommodity flows, of matrix games in sublinear time, and of block-angular convex programming problems, establishing a link to the work of Dantzig and Philip Wolfe on the decomposition principle. Indeed, one of their papers is entitled "Coordination

Complexity of Parallel Price-Directive Decomposition" [2].

In [1], Michael Fredman and Khachiyan established the surprising result that there is a quasi-polynomial-time algorithm for testing the duality of monotone disjunctive normal forms. This work had many applications, and led to a number of papers with Endre Boros, Vladimir Gurvich, his student Khaled Elbassioni, and others on various topics in combinatorics: hypergraphs, polymatroids, matroids, and enumerating all minimal solutions of implicitly stated monotone systems, with applications in minimal hypergraph traversals, data mining, machine learning, reliability theory, and in integer and stochastic programming. Finally, I want to mention his work with his student Lorant Porkolab. Porkolab and Khachiyan extended Leo's earlier work on convex polynomial programming to consider much more general formulae in the first-order theory of the reals and obtain related results. One consequence of their work [8] is that testing the feasibility of an inequality-constrained semidefinite programming problem in real or integer matrices of fixed dimension can be performed in polynomial time.

Let me conclude by telling a couple of stories that illustrate Leo's humor and sharp wit. Leo was very modest and kind to his friends, but he was also extremely cynical about politics and intolerant of condescension and pomposity. Since he had received the Young Investigators Award in Science and Technology, the Party Secretary at the Computing Center at the USSR Academy of Sciences indicated that it might be good for him to join the Party. Leo explained that he replied, with all innocence, "What party?" and added that he thought that was the right response. Later, he was looking at houses to buy near Rutgers, and was being shown around by a real estate agent, who was obviously trying to empathize as much as possible. She indicated that one house she showed him was close to the local synagogue, since she knew many Russian immigrants were Jewish, but Leo said he wasn't Jewish. Somewhat flustered, she said there were many churches close by. Leo saw his opening, and replied, "Actually, all I really believe in is the Communist Party." This caused some consternation, until finally the realtor saw a way to form a bond: "Well, they had some good ideas at the beginning."

Leonid Khachiyan was a great scholar and a much-loved father, husband, and friend. He will be sorely missed.

REFERENCES

[1] M. L. Fredman and L. G. Khachiyan, *On the complexity of dualization of monotone disjunctive normal forms*, J. Algorithms, 21 (1996), pp. 618–628.

[2] M. D. Grigoriadis and L. G. Khachiyan, *Coordination complexity of parallel price-directive decomposition*, Math. Oper. Res., 21 (1996), pp. 321–340.

[3] L. G. Khachiyan, *A polynomial algorithm in linear programming*, Doklady Akademiia Nauk SSSR, 224 (1979), pp. 1093–1096. (English Translation: Soviet Mathematics Doklady, 20 (1979), pp. 191–194.)

[4] L. G. Khachiyan, *Polynomial algorithms in linear programming*, Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki, 20 (1980), pp. 51–68. (English Translation: USSR Computational Mathematics and Mathematical Physics, 20 (1980), pp. 53–72.)

[5] L. G. Khachiyan, *Convexity and complexity in polynomial programming*, in *Proceedings of the International Congress of Mathematicians, Warsaw*, PWN, Warsaw, (1984), pp. 1569–1577.

[6] L. G. Khachiyan, *Rounding of polytopes in the real number model of computation*, Math. Oper. Res., 21 (1996), pp. 307–320.

[7] E. L. Lawler, *The great mathematical Sputnik of 1979*, The Sciences, 1980, pp. 12–15.

[8] L. Porkolab and L. G. Khachiyan, *On the complexity of semidefinite programs*, J. Global Optim., 10 (1997), pp. 351–365.

[9] S. P. Tarasov, L. G. Khachiyan, and I. I. Erlikh, *The method of inscribed ellipsoids*, Soviet Mathematics Doklady, 37 (1988), pp. 226–230.

Algebraic Methods for Integer Programming

Introduction by the Guest Editor

While linear programming based methods, such as branch-and-bound, have dominated integer programming, especially computational integer programming, we have recently seen a variety of new methods being explored. Some of these methods have been surprisingly efficient on problem instances that are difficult for branch-and-bound. In this issue we will present summaries of three recent lines of research in integer programming. First Dan Bienstock describes new results on lift-and-project methods. This theory dates back to the 1990s where Sherali and Adams, and Lovász and Schrijver provided the first corner stones. Bienstock, together with Zuckerberg and Ozbay, have developed new stronger variants of lift-and-project. Next, Kevin Woods and Ruriko Yoshida give an overview of how to use rational function representations of integer feasibility and optimization problems. This idea was pioneered by Barvinok and led to the well-known result that counting integer points in rational polyhedra can be done in polynomial time in fixed dimension. Recently, similar algorithms have been developed and also implemented with encouraging results, see the LattE home page http://www.math.ucdavis.edu/˜latte. These results are also reviewed by Woods and Yoshida. The last contribution is by Aardal, who describes how lattice basis reduction can be used in integer programming. An emphasis is made on Lenstra's algorithm, but other results, both theoretical and computational, are reviewed briefly.

Karen Aardal, June 2005.

Second Generation Lift-And-Project Algorithms

Daniel Bienstock
Department of Industrial Engineering
and Operations Research,
Columbia University, USA (dano@columbia.edu).

Abstract: We briefly describe some recent contributions to the theory of lift-and-project algorithms for 0/1 integer programming.

1. Lift-and-project algorithms

Lift-and-project methods for 0/1 integer programming proceed by reformulating a given integer program through the addition of (potentially many) new variables and constraints. The objective of the reformulation is to make the combinatorial structure of a problem more explicit — even though the reformulation does not change the underlying combinatorial problem, the resulting continuous relaxation is tighter due to the richness of the new variables and constraints.

This is the 'lift' phase. The 'project' component of the algorithms, not always present, concerns the efficient solution of the "lifted" formulation by projection back to the original space of variables. In the case of linear reformulations, this amounts to an appropriate application of the Farkas' lemma.

The initial motivation for such algorithms lies in the observation that, frequently, given a polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$, if $\mathcal{Q} \subseteq \mathbb{R}^N$ (for some $N > n$) is such that $\mathcal{P}$ is the projection of $\mathcal{Q}$ to $\mathbb{R}^n$, then the polyhedral structure of $\mathcal{Q}$ is simpler than that of $\mathcal{P}$.

Our recent work [3], [4] provides new lifting algorithms with provably stronger guarantees. The purpose of this note is to place these results in the context of prior algorithms.

2. The Sherali-Adams and Lovász-Schrijver operators

Sherali and Adams [16] and Lovász and Schrijver [12] described the first formal reformulation methodologies. Here we will describe a single lifting algorithm that constitutes an amalgam of ideas from [16] and [12].

To describe this algorithm, consider the feasible set

$$\mathcal{F} \;=\; \{\, x \in \{0,1\}^n \;:\; Ax \geq b \,\} \tag{1}$$

for a 0/1 integer program, where A is a matrix and b is a vector of appropriate dimensions. Let $1 \leq t \leq n$ be a fixed integer, and consider a lifted formulation using the 0/1 variables $v[Y,N]$, for all pairs of disjoint Y, $N \subseteq \{1,2,\ldots,n\}$ with $|Y \cup N| \leq \min\{t+1, n\}$. The interpretation of these variables is that $v[Y,N] = 1$ if and only if

$$x_j = 1, \text{ for all } j \in Y, \text{ and } x_j = 0, \text{ for all } j \in N. \tag{2}$$

Now, (2) is a logical statement — the challenge is how to approximate it with, for example, linear inequalities. This is one of the tasks undertaken by the lifting algorithms, which furthermore impose on the v variables additional constraints implied by $Ax \geq b$. As examples of inequalities that (2) implies, we have

$$v \geq 0, \qquad v[\emptyset,\emptyset] = 1, \tag{3}$$

$$v[Y \cup j, N] + v[Y, N \cup j] \;=\; v[Y,N], \quad \forall\, j \tag{4}$$

for all appropriate Y, N. Note that (4) amounts to the familiar inclusion-exclusion principle of combinatorics. Further, consider row h of $Ax \geq b$; namely $\sum_i a_{hi} x_i \geq b_h$. Then for any disjoint Y, N with $|Y \cup N| \leq t$, we must also have:

$$\sum_j a_{hi}\, v[Y \cup i, N] \;-\; b_h\, v[Y,N] \;\leq\; 0. \tag{5}$$

In order to see that (5) is valid, first note that (4) implies that $v[Y,N] = 0$ whenever $Y \cap N \neq \emptyset$ (proof by induction on $|Y \cap N|$). Suppose that $x \in \mathcal{F}$. If $v[Y,N] = 0$, then by definition all terms on the left-hand side of (5) are zero as well. And if $v[Y,N] = 1$, then for any $1 \leq i \leq n$, $x_i = v[Y \cup i, N]$ as desired.

For given $1 \leq t \leq n$, the collection of inequalities (3)–(5) is known as the level-t Sherali-Adams reformulation; and the procedure that generates the reformulation is denoted by SA^t. Note that we can "project" back to the space of the original variables by setting $x_j = v[\{j\},\emptyset]$, for $1 \leq j \leq n$. Also note that (3), (4) imply that $v[Y,N] \leq 1$ for all Y, N enumerated. It can be shown [16], also see [11] that for $t = n$ the reformulation yields an integral polyhedron.

In order to see the relationship between this operator and the Lovász-Schrijver operators [12], consider the case $t = 1$ and assume $n > 1$. Using equation (4), we can *eliminate* all $v[Y,N]$ with $N \neq \emptyset$. The remaining variables will be of the form $v[Y,\emptyset]$ with $|Y| \leq 2$. Consider the $(n+1)\times(n+1)$ matrix M with rows and columns indexed by the sets S with $|S| \leq 1$ (i.e., the singletons and the empty set) defined by:

$$M_{S,T} \;=\; v[S \cup T, \emptyset],$$

for all S, T with cardinality ≤ 1. Then:

(LS.1) M is symmetric; its $(\emptyset,\emptyset)$-entry equals 1 (this last property follows from $v[\emptyset,\emptyset] = 1$).

(LS.2) For any S with $|S| \leq 1$, $M_{S,S} = M_{\emptyset,S} = M_{S,\emptyset}$; for any singletons S, T, $M_{S,T} \leq M_{S,\emptyset} \leq 1$.

For convenience, let us order the rows of M so that the $\emptyset$-row is the zeroth (and similarly with the columns); denote the entries of M by m_{ij}, where $0 \leq i, j \leq n$ with the obvious meaning. Then e.g. the last condition can be restated as $m_{0i} = m_{i0} = m_{ii}$ for all $0 \leq i \leq n$.

Consider an arbitrary row h of $Ax \geq b$ and let $1 \leq j \leq n$. The inequality obtained from (5) using $Y = \{j\}$ and $N = \emptyset$ is then:

$$\sum_{i \geq 1} a_{hi} m_{ij} \;-\; b_h m_{\emptyset,j} \;\;\geq\;\; 0. \tag{6}$$

In other words:

(LS.3) column j of M satisfies each constraint of $Ax \geq b$, in homogenized form.

Suppose now that we consider (5) using $Y = \emptyset$ and $N = \{j\}$. Then we obtain

$$\sum_{i \geq 1} a_{hi} v[\{i\},\{j\}] \;-\; b_h v[\emptyset,\{j\}] \;\;\geq\;\; 0 \tag{7}$$

(comment: summing over all j is correct since $v[\{j\},\{j\}] = 0$ by (4)). But using (4, $v[\{i\},\{j\}] = v[\{i\},\emptyset] - v[\{i,j\},\emptyset]$ for each $1 \leq i \leq n$ (and similarly $v[\emptyset,\{j\}] = v[\emptyset,\emptyset] - v[\{i\},\emptyset]$); substituting in (7) now yields:

$$\sum_{i \geq 1} a_{ij}(m_{i0} - m_{ij}) \;-\; b_h(m_{00} - m_{j0}) \;\;\geq\;\; 0. \tag{8}$$

In other words:

(LS.4) the vector obtained by subtracting column j of M from the zeroth column also satisfies each constraint of $Ax \geq b$, in homogenized form.

Requirements **(LS1)-(LS4)** constitute the Lovász-Schrijver operator N (or, more precisely, "projecting" by setting $x_j = m_{j0}$ for $1 \leq j \leq n$ describes the operator), what we have done here is to show that this is equivalent to the SA^1 operator.

An additional and independent insight is that conditions **(LS.1) and (LS.2)** will hold if matrix M is of the form ww^T, for some 0/1-vector w. This is certainly the case if x is a 0/1-vector, in which case $w^T = (1x^T)$. But what additional implications can be imposed, that are consistent with $M = ww^T$? One such condition is:

(LS.5) M is positive-semidefinite.

By imposing this condition, in addition to **(LS.1)-(LS.4)** one obtains the Lovász-Schrijver operator N_+.

2.1 An extended operator

The matrix-oriented approach described in the previous section can be extended to arbitrary $1 < t \leq n$. Namely, one defines a matrix M, with rows and columns indexed by all subsets of variables with cardinality $\leq t$. If we have run the level-$2t$ Sherali-Adams operator, with vector v, then we would set $M_{S,T} = v[S \cup T, \emptyset]$. But we do not strictly need to run the operator and still directly impose appropriate constraints on M. These constraints are generalizations of **(LS.1)-(LS.4)**. More precisely, **(LS.1)** is unchanged, but a more general version of **(LS.2)** is needed (namely: the $\emptyset$-row and -column are equal to the main diagonal, and $M_{S,T} \leq M_{S',T'}$ if $|S \cup T| \leq |S' \cup T'|$). Constraints **(LS.3)-(LS.4)** remain unchanged.

However, there is something more substantial that can be said with regards to **(LS.2)**. Namely, we can impose that $M_{S,T} = M_{S',T'}$ whenever $|S \cup T| = |S' \cup T'|$. The point is that this condition goes far beyond the symmetry requirement of the Lovász-Schrijver operator, and that it really becomes effective for $t > 1$.

With regards to the Sherali-Adams operator, the conditions we just described yield an operator that lies "between" the level-t and the level-$2t$ operators in terms of the constraints that it imposes. Finally, we can also impose positive semidefiniteness on M.

The procedure we just described is meant as a simple example of how the algebra of subsets of variables can be used to produce more general procedures than the SA^t, N and N_+ operators.

3. Subset algebra operators

In this section we describe the approach used in [3], [18]. In order to describe the core of this method, consider an arbitrary statement that could be made about a 0/1 vector x. For example:

$$\text{K:} \quad \text{“}\textstyle\sum_j x_j \text{ is a prime number”}.$$

Now, corresponding to this statement we could introduce a new 0/1 variable, $v[\text{K}]$, with the intention that

$$v[\text{K}] = 1, \quad \text{precisely when K holds.} \tag{9}$$

Of course, (9) is a purely logical statement and the challenge would be to approximate it by imposing constraints involving variable $v[\text{K}]$ that can be cast in the language of convex optimization.

In our method we proceed in two steps. First, we use statements that are described by set-theoretic expressions, and are somewhat more pliable than the example of K given above. At the same time, the statements are amenable to effective approximation by linear inequalities, while providing sharp information on the underlying combinatorial structure of the set $\mathcal{F}$. Second, having chosen a set of new variables $v[\text{K}]$ we will then proceed by creating a "matrix of variables" M as in Section 2.1.

Set covering

The first operator is simplest to describe in the context of set covering problems, that is to say when the matrix A in (1) is 0/1 and b is the vector of 1s. Borrowing notation from Section 2., for each $1 \leq j \leq n$, let Y_j denote the expression "$x_j = 1$" (this is a set-theoretic expression in that it describes a subset of the n-hypercube). Similarly, let N_j denote the expression "$x_j = 0$". Finally, a set S of column indices is called an *intersection of supports* if, for some pair i, i' of rows of A, $S = \{1 \le k \le n \,:\, a_{ik} = 1 \text{ and } a_{i'k} = 1\}$.

The operator we describe creates a collection of variables corresponding to each intersection of supports S. Each of these variables is indexed by a set-theoretic expression:

(S.0) $\alpha_S = \cap_{j \in S} N_j$,

(S.1) $\beta_S^k = (\cap_{j \in S-k} N_j) \cap Y_k$, for each $k \in S$, and

(S.2) τ_S, where τ_S is the set-theoretic expression that states "$\sum_{j \in X} x_j \ge 2$".

As discussed above, we stress that the definitions in (S.0)-(S.2) give the *intended* meaning for the new variables — we have to introduce constraints that are consistent with this meaning. Further, from an algorithmic standpoint we want to use the symbol τ_S, rather than the exact set-theoretic formula which is straightforward, but of exponential length in terms of the N_j and Y_j:

$$\cup \{\, (\cap_{j \in P} Y_j) \cap (\cap_{j \in S-P} N_j) \,:\, P \subseteq S,\ |P| \ge 2 \,\}.$$

In addition, for each intersection of supports S, we introduce, for each $h \notin S$, the variable indexed by the expression $Y_h \cap \alpha_S$; for each $k \in S$ and each $h \notin S$ the expression $Y_h \cap \beta_S^k$; and for each $1 \le h \le n$, the expression $Y_h \cap \tau_S$.

In total, thus, when the matrix A has m rows there are $O(m^2 n^2)$ variables. The constraints that we impose are the following. Let S be an intersection of supports. Corresponding to S, we have

$$v[\alpha_S] + \sum_{k \in S} v[\beta_S^k] + v[\tau_S] = 1 \tag{10}$$

$$v[Y_h \cap \alpha_S] + \sum_{k \in S} v[Y_h \cap \beta_S^k] + v[Y_h \cap \tau_S] = v[Y_h],\ \forall\, h \notin S, \tag{11}$$

$$v[\beta_S^h] + v[Y_h \cap \tau_S] = v[Y_h],\ \forall\, h \in S, \tag{12}$$

$$\sum_{h \in S} v[Y_h \cap \tau_S] \ \ge\ 2\, v[\tau_S]. \tag{13}$$

The first three constraints state that $\sum_{j \in S} x_j$ must equal 0, 1, or at least 2; (13) makes an additional (obvious) statement regarding the ≥ 2 case. Furthermore, for each row r of A, we impose the homogenizations:

$$\sum_{h \notin S} a_{rh} v[Y_h \cap \alpha_S] \ \ge\ v[\alpha_S], \tag{14}$$

$$\sum_{h \notin S} a_{rh} v[Y_h \cap \beta_S^k] + a_{rk} v[\beta_S^k] \ge v[\beta_S^k],\ \forall\, k \in S, \tag{15}$$

$$\sum_{h} a_{rh} v[Y_h \cap \tau_S] \ \ge\ v[\tau_S]. \tag{16}$$

The last set of constraints is

$$\sum_{h} a_{ih} v[Y_h] \ \ge\ 1, \quad \text{for each row } i \text{ of } A. \tag{17}$$

The number of constraints is $O(m^3 n)$. This concludes the description of the formulation (the variables are assumed nonnegative and upper bounded by 1). We have:

Lemma 3..1 *Consider any feasible solution $\hat{v}$ to the system just described. Then the vector with entries $\hat{v}[Y_h]$, $1 \le h \le n$, satisfies each inequality valid for $\mathcal{F}$ with coefficients 0, 1 or 2.*

Proof Sketch. For simplicity, consider an inequality valid for $\mathcal{F}$ of the form:

$$\sum_{h \in T} x_h \ \ge\ 2, \tag{18}$$

for some set T. Since (18) is valid for $\mathcal{F}$, there are two rows i, i' of A whose supports are contained in T. Let i' be as desired; without loss of generality the supports of i and i' are not disjoint, or else (18) is trivially satisfied by (17).

Let S be the intersection of the supports of i, and i'. Now if we can argue that $\hat{v}$ satisfies:

$$\sum_{h \in T-S} \hat{v}[Y_h \cap \alpha_S] \ \ge\ 2\, \hat{v}[\alpha_S], \tag{19}$$

$$\sum_{h \in T-S} \hat{v}[Y_h \cap \beta_S^k] + v[\beta_S^k] \ge 2\, \hat{v}[\beta_S^k],\ \forall k \in S, \tag{20}$$

$$\sum_{h \in T-S} \hat{v}[Y_h \cap \tau_S] \ \ge\ 2\, \hat{v}[\tau_S], \tag{21}$$

then by (10 - 12) we will be done. Note that we already have (21): it is (13). (19) follows by applying (14) to rows i and i' and adding. Finally, pick any $k \in S$.

Since (18) is valid for $\mathcal{F}$, there is another row i'' of A whose support is contained in T, and such that $a_{i''k} = 0$. As a result, applying (15) with $r = i''$ yields (20), as desired. ■

Two characteristics differentiate the reformulation operator described from the SA^t, N and N_+: first, the new variables depend on the structure of the constraints, and second, the new variables are indexed by set-theoretic expressions far more complex than simple conjunctions of Y_j and N_j terms. With regards to the second point, note that, for example, an α_S expression involves S terms; and furthermore in order to describe a τ_S expression using conjunctions of the Y_j and N_j we would need almost $2^{|S|}$ terms.

The operator given above can be significantly generalized. For this we need a new definition:

Definition 3..2
(a) *The* **pitch** *of an inequality $\alpha^T x \geq \beta$ with non-negative coefficients is the smallest integer k, such that the sum of the k smallest nonzero α_j is at least β.*
(b) *Given a feasible region (1), we denote by $P^k_{A,b}$ the polyhedron defined by all valid inequalities of pitch $\leq k$.*

Thus, an inequality with coefficients of value (say) $0, 1, 2, 3$ or 4 has pitch ≤ 4. The following result is proved in [3]:

Theorem 3..3 *[3] For each integer $k \geq 2$ there is an integer g_k with the following property. Given a set-covering problem (1) with m rows and n columns, there is a polyhedron $\mathcal{L}_k$ such that:*

(i) $\mathcal{L}_k$ is described by a system of $O((n+m)^{g_k})$ inequalities in $O((n+m)^{g_k})$ variables, that is computable in time $O((n+m)^{g_k})$, and whose coefficients are integral and with absolute value at most k,

(ii) $\mathcal{F}_{A,e} \subseteq \pi(\mathcal{L}_k) \subseteq P^k_{A,e}$.

3.1 Lift-and-project and Chvátal-Gomory rank.

Here we outline some of the theoretical impact of Theorem 3..3. This concerns, in particular, the comparison between the relaxation implicit in the formulation provided by Theorem 3..3, and Chvátal-Gomory closures of the continuous relaxation of $\mathcal{F}$ of fixed rank.

We remind the reader of the definition of the Chvátal-Gomory cuts. Given $\mathcal{F}$, where A has m rows, denote by $\mathcal{R}$ the continuous relaxation of $\mathcal{F}$. Suppose we choose multipliers $\pi_i \geq 0$ for $1 \leq i \leq m$, and multipliers $0 \leq \gamma_j$ for $1 \leq j \leq n$. Then the following inequality is valid for $\mathcal{F}$

$$\sum_j \lceil \pi^T A - \gamma_j \rceil x_j \quad \geq \quad \left\lceil \pi^T b - \sum_j \gamma_j \right\rceil , \quad (22)$$

and is denoted a rank-≤ 1 inequality. The set of all such inequalities, together with the bounds $0 \leq x_j \leq 1$ defines the rank-1 closure of $\mathcal{R}$. Using the rank-1 closure of $\mathcal{F}$ to generate more valid inequalities in a similar way, one then obtains the rank-2 closure of $\mathcal{R}$, and so on. The rank of a valid inequality for $\mathcal{F}$ is the smallest integer q such that the rank-q closure implies the inequality — a classical result is that each valid inequality for $\mathcal{F}$ has rank that depends on n only. Furthermore, many commercial integer programming systems produce low-rank cuts, while at the same time separating over the rank-1 closure of an arbitrary set $\mathcal{R}$ is NP-hard (also see [8], [5]).

To consider how Theorem 3..3 relates to the rank-1 closure of a set covering problem, consider an inequality (22) that is not dominated. If its right-hand side is k (say), then its pitch is $\leq k$, and so the formulation produced by Theorem 3..3 satisfies (22). On the other hand, if the right-hand side is "large" (greater than k) then the continuous relaxation of $\mathcal{F}$ satisfies (22) within additive error ≤ 1, and thus within relative error $\leq \frac{1}{k}$. Thus, by appropriately choosing k, we will satisfy all rank-≤ 1 inequalities within arbitrarily small tolerance while still using a polynomial-size formulation.

In fact, something much stronger can be proved. The issue here is that if we have polyhedra of the form $\mathcal{A}' = \{A'x \geq b', \, x \in [0,1]^n\}$ and $\mathcal{A}'' = \{A''x \geq b'', \, x \in [0,1]^n\}$ such that whenever $x \in \mathcal{A}''$ then $A'x \geq (1-\epsilon)b'$ (ϵ small), it may still be the case that $\min\left\{c^T x : x \in \mathcal{A}''\right\}$ is much smaller than $\min\left\{c^T x : x \in \mathcal{A}'\right\}$. Thus, if both $\mathcal{A}'$ $\mathcal{A}''$ are relaxations of the same set integer program, $\mathcal{A}'$ may still be far stronger even though (seemingly) $\mathcal{A}''$ closely approximates $\mathcal{A}'$ (the reason behind this situation is that the x_j are upper bounded by 1 — this can be used to generate small examples of such $\mathcal{A}'$, $\mathcal{A}''$).

However, this pathological behavior *cannot happen* when $\mathcal{A}'$ is the rank-1 closure of a set $\mathcal{R}$. This is due to the structure of rank-1 cuts: if $\sum_j \alpha_j x_j \geq \beta$ is a rank-≤ 1 cut, then so is $\sum_{j \neq k} \alpha_j x_j \geq \beta - \alpha_k$, for any given $1 \leq k \leq n$ (or, more precisely, it is implied by inequalities defining the rank-1 closure). The following theorem is proved in [4].

Theorem 3..4 *For each integer $q > 0$ and $0 < \epsilon < 1$ there are integers $d = d(q,\epsilon)$ and $u = u(q,\epsilon)$ with the following property. Let A be an $m \times n$ nonnegative matrix, and $b \in \mathbb{R}^m_+$. Assume we are given the set of minimal covers arising from each row of $Ax \geq b$, and let $\nu_{A,b}$ denote the number of such covers. Then, in time $O((n+m+\nu_{A,b})^d)$ we can compute a formulation described by $O\left((n+m+\nu_{A,b})^d\right)$ rows and columns, with coefficients that are integral and with absolute value at most u, and whose feasible region $\hat{\mathcal{R}}$ satisfies*

(i) $\mathcal{F} \subseteq \pi(\hat{\mathcal{R}})$,

(ii) for any $c \in \mathbb{R}^n_+$, $\min\{c^T x : x \in \pi(\hat{\mathcal{R}}) \cap \mathcal{R}_{A,b}\} \geq (1-\epsilon)\tau^{(q)}_{A,b}(c)$.

The formulation described by the theorem is precisely that given by Theorem 3..3. We refer the reader to [4] for details. Note that when (1) is a set-covering problem, then the minimal covers are defined by rows of A. In the set-covering case, therefore, Theorem 3..4 says that in polynomial-time we can approximate arbitrarily closely the *strength* of a fixed-rank closure, thereby avoiding the pathological situation described above.

A good example is that provided by a set-covering problem defined by a full-circulant matrix:

$$\mathcal{F} \;=\; \{\, x \in \{0,1\}^n \;:\; \sum_{j \neq k} x_j \geq 1, \quad \text{for each } 1 \leq k \leq n \,\}, \tag{23}$$

and the optimization problem $v^* = \min\left\{\sum_j x_j : x \in \mathcal{F}\right\}$. Clearly, $v^* = 2$. By Theorem 3..3 we have a polynomial-size formulation that proves $v^* = 2$. In contrast to this, in [4], [3] the following is shown:

1. For each fixed $1 \leq t$, the Sherali-Adams level-t SA^t operator, and the Lovász-Schrijver t-iterated operator N^t_+ can **only** prove $v^* \geq 1 + \frac{t+1}{n} + o(\frac{1}{n})$ (i.e. an underestimate by nearly a factor of 2).

2. The valid inequality $\sum_j x_j \geq 2$ has rank $\geq n-3$ for both the SA and N_+ operators (i.e. it requires exponential time to prove the inequality using SA or N_+).

In fact, similar negative results can be shown for a much stronger operator than either SA or N_+. The weakness of the operators stems from the use of simple "monomials" with up to t variables when forming the new variables for a lifted formulation. In a sense this approach is too "pedantic" and it simply does not adequately capture the combinatorial structure of the problem. We expect that a similar result may well hold for the Lasserre operator in [10], for similar reasons. For earlier negative results, see [9], [11], [7].

Theorem 3..4 requires that the set of minimal covers for a problem 1 be given explicitly. What can be said if that is not the case? Note that the problem of detecting whether some minimal cover is violated by a fractional vector $\hat{x}$ is (weakly) NP-hard (see [15] for background). Still, in the context of an approximation result like Theorem 3..4 this is a difficulty that can probably be circumvented. A more significant obstacle is that the number of minimal covers may be exponentially large. In order to deal with such a difficulty, one would have to extend our subset-algebra lifting procedure so as to handle minimal covers implicitly, via the ellipsoid method. A result along these lines is provided in the online version of [4].

3.2 Packing problems, tree-width, and the Sherali-Adams operator

The operators and results described above concern covering-type problems, i.e. problems (1) where $A \geq 0$. What can be said about packing-type problems, that is feasible regions of the form

$$\mathcal{F} \;=\; \{\, x \in \{0,1\}^n \;:\; Ax \leq b \,\} \tag{24}$$

where $A \geq 0$? Ideally, a result such as Theorem 3..3 would apply here, as well, paving the way to a result like Theorem 3..4.

Up to date, the best we have been able to do in this regard is to prove a structural result concerning the strength of the Sherali-Adams operator. This result involves two concepts.

One is that of *tree-width.* For an integer w, a graph G is said to be of tree-width $\leq w$ if G is contained in a chordal graph H such that the clique number of H is $\leq w + 1$. This definition, while succinct, does not really do justice to the richness of this concept — see [17] and [2] (and references therein) for additional material. A simple way to view tree-width is that the higher the tree-width of a graph, the higher its "complexity", from a fundamental perspective.

The second concept is that of *clique graph.* Given a matrix A, its clique graph has a vertex for every column, and an edge between two vertices j_1 and j_2 if there exists some row i with $a_{i,j_1} \neq 0$ and $a_{i,j_2} \neq 0$. (Again, this is an old concept, refer to [15]).

Armed with these two concepts, we can define the tree-width of a valid inequality $\sum_j \alpha_j x_j \geq \beta$ for a packing set (24): this is the minimum tree-width of the clique graph of any row-submatrix A' of A, such that $\sum_j \alpha_j x_j \geq \beta$ is valid for $\{ x \in \{0,1\}^n \ : \ A'x \leq b' \}$. Here, A' is obtained from A by removing some rows, and correspondingly for b'. The main result proved in [2] is:

Theorem 3..5 *Consider a packing set $\mathcal{F}$ as in (24). Let $k \geq 1$, and suppose that a vector $\hat{x} \in \mathbb{R}^n$ satisfies the constraints imposed by the level-k Sherali–Adams operator applied to $\mathcal{F}$. Then*

(1) $\hat{x}$ satisfies every valid inequality $\alpha^T x \leq \beta$ whose tree-width is at most $k - 1$.

(2) Suppose A is 0/1 and b is integral. Then $\hat{x}$ satisfies every valid inequality $\alpha^T x \leq \beta$ whose tree-width is at most $\min\{k, n-1\}$.

As a result, in polynomial-time the Sherali-Adams operator produces a formulation whose solutions are guaranteed to satisfy a number of classical inequalities for vertex-packing problems, such as the odd-hole and odd-wheel inequalities. This result was already known (it is essentially given in [12]). But Theorem 3..5 is stronger than this, and it covers far more complex inequalities, such as antiweb-wheels [6]. In essence tree-width provides a means for parameterizing the "complexity" of a valid inequality for packing problems.

Note that odd-holes, odd-wheels, and the antiweb-wheels (and other inequalities) that can be separated in polynomial time in [6] all have bounded Chvátal-Gomory rank. We conjecture that the techniques used to obtain Theorem 3..5, perhaps applied to a stronger operator than Sherali-Adams (possibly using positive-semidefiniteness) will yield results for packing problems similar to those we described above for covering problems, mainly the polynomial-time approximation of fixed-rank Chvátal-Gomory closures.

3.3 Computation

Of course, a critical research goal would be to effectively implement the subset algebra lifting operator described above. Using today's computing machinery this would be quite difficult because of the large size of the resulting formulations. An interesting possibility would be to experiment with classical techniques such as column generation and Lagrangian relaxation — but we feel that at least some theoretical understanding of *how* to apply such techniques in this context would be needed, since the formulations we produce are very highly structured.

At the same time, we strongly feel that the use of ever more powerful convex relaxations of 0/1 integer programs is likely to produce good results in the future. On the one hand, this approach leverages the very robustly growing field of convex optimization, which is backed up by strong theory and very careful computation. There is also an equally robust concurrent effort at adapting tools from nondifferentiable optimization to approximate the solution of very large structured linear and convex programs (see, e.g., [14], [1], [13]). Finally, computational machinery continues to improve, especially with regards to memory. All of this leads us to believe that our ability to solve large convex optimization problems will dramatically improve over the next decade; combined with effective reformulation procedures this should lead to much better techniques for proving stronger bounds for 0/1 integer programs.

We would contrast this with the "classical" approach of attempting to apply branch-and-cut directly to a formulation. For difficult combinatorial problems this is likely to generate an excessive amount of enumeration; i.e., the algorithms will not scale with problem size no matter how much computational power is available.

REFERENCES

[1] D. Bienstock and G. Iyengar, *Solving fractional packing problems in O*(1/epsilon) iterations*, *Proc. 26th Ann. Symp. Theory of Computing* (Chicago, 2004), pp. 146–155 (CORC report TR-2003-03).

[2] D. Bienstock and N. Ozbay, *Tree-width and the Sherali-Adams operator*, Discrete Optimization, 1 (2004), pp. 13–22.

[3] D. Bienstock and M. Zuckerberg, *Subset algebra lift operators for 0-1 integer programming*, SIAM J. Optim., 15 (2004) pp. 63–95.

[4] D. Bienstock and M. Zuckerberg, *Approximate fixed-rank closures of covering problems*, CORC report TR-2003-01, Math. Program., to appear.

[5] A. Caprara and A. Letchford, *On the separation of split cuts and related inequalities*, Math. Program., 94 (2003), pp. 279–294.

[6] E. Cheng and S. de Vries, *Antiweb-wheel inequalities and their separation problems over the stable set polytopes*, Math. Program., 92 (2002), pp. 153–175.

[7] W. Cook and S. Dash, *On the matrix-cut rank of polyhedra*, Math. Oper. Res., 26 (2001), pp. 19–30.

[8] G. Cornuéjols and Y. Li, *A connection between cutting plane theory and the geometry of numbers*, Math. Program., 93 (2002), pp. 123–127.

[9] M.X. Goemans and L. Tunçel, *When does the positive semidefiniteness constraint help in lifting procedures*, manuscript (2000).

[10] J. B. Lasserre, *An explicit exact SDP relaxation for nonlinear 0-1 programs*, in *Lecture Notes in Computer Science*, K. Aardal and A. M. H. Gerards, editors, (2001), pp. 293-303.

[11] M. Laurent, *A Comparison of the Sherali-Adams, Lovász-Schrijver and Lasserre relaxations for 0-1 programming*, Math. Oper. Res., 28 (2003), pp. 470–496.

[12] L. Lovász and A. Schrijver, *Cones of matrices and set-functions and 0-1 optimization*, SIAM J. Optim., 1 (1991), pp. 166–190.

[13] A. Nemirovski, *Prox-method with rate of convergence O(1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems*, SIAM J. Optim., 15 (2004), pp. 229–251.

[14] Y. Nesterov, *Smooth minimization of non-smooth functions*, Math. Program., 103 (2005), 127 - 152.

[15] G.L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[16] S. Sherali and W. Adams, *A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems*, SIAM J. Discrete Math., 3 (1990), pp. 411–430.

[17] N. Robertson and P. D. Seymour, *Graph minors II: Algorithmic aspects of tree-width*, J. Algorithms, 7 (1986), pp. 309–322.

[18] M. Zuckerberg, *A Set Theoretic Approach to Lifting Procedures for 0,1 Integer Programming*, Ph.D. Thesis, Columbia University, 2004.

Short Rational Generating Functions and Their Applications to Integer Programming

Kevin Woods
Department of Mathematics, University of California,
Berkeley, CA 94720-3840, USA
(kwoods@math.berkeley.edu).

Ruriko Yoshida
Department of Mathematics, Duke University,
Durham, NC 27708-0320, USA (ruriko@math.duke.edu).

Abstract: In this paper, we provide an overview of rational generating function tools for encoding integer points inside a rational polyhedron, and we examine their applications to solving integer programming problems.

1. Introduction

In the 1980's, H.W. Lenstra, Jr. developed an algorithm to *detect* integer points in a rational polyhedron using the LLL-algorithm [17, 20], and this algorithm runs in time polynomial in the input size if we fix the dimension. Lenstra used this algorithm to show that integer programming problems with a fixed number of variables can be solved in polynomial time. A later algorithm of similar structure, by Lovász and Scarf [21], was implemented by Cook, et al. [9]. In addition, Aardal and collaborators [1, 2, 3] have written fairly effective modifications of the LLL-procedure for testing integer feasibility. In the 1990's, based on work by the geometers Brion, Khovanski, Lawrence, and Pukhlikov, Barvinok discovered an algorithm to *count* integer points in rational polytopes, and this algorithm also runs in polynomial time if we fix the dimension [5, 6]. The idea of the algorithm is to encode all the integer points inside a rational polyhedron into a *rational generating function.* In particular, if $P \subset \mathbb{R}^d$ is a rational polyhedron, define the generating function

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

where $\mathbf{x}^s$ denotes $x_1^{s_1} \cdots x_d^{s_d}$ with $s = (s_1, \ldots, s_d)$. After computing $f(P;\mathbf{x})$ as a rational function, we evaluate $|P \cap \mathbb{Z}^d| = f(P;1,\ldots,1)$.

Example 1..1 *Suppose P is the one-dimensional polytope $[0,N]$. Then $f(P;x) = 1+x+x^2+\cdots+x^N$, $f(P;x)$ can be represented by the rational function $\frac{1-x^{N+1}}{1-x}$, and $f(P;1) = N+1$, the number of integer points in P.*

Note that the rational function representation of $f(P;\mathbf{x})$ in Example 1..1 is "short". In general, the number of terms in the representation will be bounded by a polynomial in the input size. Also note that substituting $x = 1$ yields a denominator equal to zero in the rational function, so some analytic technique must be used to evaluate $f(P;1)$. In this particular case, we could take the limit as x approaches 1 and apply l'Hospital's rule. In general, we must use more complicated residue calculus [5].

Shortly after Barvinok introduced his algorithm, Dyer and Kannan [16] showed that a particular step of his counting algorithm, which originally relied on Lenstra's result, could be replaced by a short-vector computation using the LLL-algorithm. This result gives a new proof that integer programming problems with a fixed number of variables can be solved in polynomial time: using binary search, one can turn Barvinok's counting oracle into an algorithm that solves integer programming problems. This algorithm was proposed by Barvinok in [6]. We call this integer programming algorithm the *BBS algorithm.*

In this report, we will provide a brief overview of how to encode the integer points inside a given polyhedron into a short rational generating function (Section 2.), and then we will survey applications to solving integer programming problems (Section 3.).

2. Computing short rational generating functions

In 1993, Barvinok [5] gave an algorithm that, given a rational polyhedron $P \subset \mathbb{R}^d$, computes $f(P;\mathbf{x})$ as a rational function in polynomial time (when the dimension of the polyhedron is fixed). In this section, we outline the steps of Barvinok's algorithm. For a more lengthy and detailed exposition, see [6].

Let $\{c_1, \ldots, c_d\}$ be a basis for the lattice $\mathbb{Z}^d$, and let $\beta_1, \ldots, \beta_d \in \mathbb{Q}$ be given. We define the *rational unimodular cone* $K = \{x \in \mathbb{R}^d : \ c_i \cdot x \leq \beta_i,$ for all $i\}$, where $a \cdot b$ is the standard dot product.

Computing the generating function of a unimodular cone is "easy". In fact, if $\{u_1, \ldots, u_d\}$ is the (negative) dual basis of $\mathbb{Z}^d$ such that

$$u_i \cdot c_j = \begin{cases} -1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases},$$

and if $v = -\sum_{i=1}^{d} \lfloor \beta_i \rfloor u_i$, then (by Lemma 4.1 of [6])

$$f(K; \mathbf{x}) = \frac{\mathbf{x}^v}{(1-\mathbf{x}^{u_1}) \cdots (1-\mathbf{x}^{u_d})}. \tag{1}$$

Example 2..1 *Suppose we have a rational unimodular cone $K \subset \mathbb{R}^2$, such that*

$$K = \{(x_1,\ x_2) \in \mathbb{R}^2 : x_1 \leq 4,\ 2x_1 + x_2 \leq 10\}.$$

In our example, $c_1 = (1,\ 0)$ and $c_2 = (2,\ 1)$ form a basis of $\mathbb{Z}^2$. Then, we have that $u_1 = (-1,\ 2)$, $u_2 = (0,\ -1)$, and $v = -[4(-1,\ 2) + 10(0,\ -1)] = (4,\ 2)$. Therefore

$$f(K; x_1, x_2) = \frac{x_1^4 x_2^2}{(1 - x_1^{-1} x_2{}^2)(1 - x_2^{-1})}.$$

The idea of Barvinok's algorithm, then, is to reduce the case of computing $f(P; \mathbf{x})$ to the case of computing the generating functions for a collection of unimodular cones. We do this in three steps: first reduce to the case of general rational cones, then to the case of simplicial cones (rational cones with exactly d extreme rays), and then finally to unimodular cones. If v is a vertex of P, then define the *supporting cone*, $\text{cone}(P, v)$, of P at v, as follows. Suppose that P is defined by $P = \{x \in \mathbb{R}^d :\ c_i \cdot x \leq \beta_i,\ \text{for } i = 1, \ldots, m\}$, for some $c_i \in \mathbb{Q}^d$, $\beta_i \in \mathbb{Q}$. For a vertex, v, of P, let $I_v = \{i :\ c_i \cdot v = \beta_i\}$, and define

$$\text{cone}(P, v) = \{x \in \mathbb{R}^d :\ c_i \cdot x \leq \beta_i,\ \text{for } i \in I_v\}.$$

Then the following theorem from [8] allows us to compute $f(P, \mathbf{x})$ by computing the generating functions of the cones $\text{cone}(P, v)$, where v is a vertex of P.

Theorem 2..2 *(Brion's Theorem) Let P be a rational polyhedron. Then*

$$f(P; \mathbf{x}) = \sum_v f\big(cone(P, v); \mathbf{x}\big),$$

where the sum runs over all vertices v of P.

Example 2..3 *Let P be the one-dimensional polytope $[0, N]$. Then $v_0 = 0$ and $v_1 = N$ are the vertices of P, $cone(P, v_0) = [0, \infty)$ and $cone(P, v_1) = (-\infty, N]$, and so*

$$f(P; x) = f(cone(P, v_0); x) + f(cone(P, v_1); x)$$
$$= \tfrac{1}{1-x} + \tfrac{x^N}{1-x^{-1}} = \tfrac{1-x^{N+1}}{1-x}.$$

Note that, since the dimension d is fixed, we may compute the vertices of P in polynomial time (see [22] for details). Next, by triangulating these cones, we reduce to the case of simplicial cones. There are efficient algorithms, when the dimension is fixed, to perform triangulations (see [4, 19] for details). Finally, let K be a simplicial cone. The essential contribution of Barvinok [5] was to show that we can decompose K into a *signed* collection of unimodular cones. To be precise, given a set $A \subset \mathbb{R}^d$, define the indicator function, $[A] : \mathbb{R}^d \to \mathbb{R}$, of A by

$$[A](x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases}.$$

Then we have the following theorem from [5].

Theorem 2..4 *Fix d. There is a polynomial time algorithm which, given a rational simplicial cone $K \subset \mathbb{R}^d$, computes rational unimodular cones K_i and signs $\epsilon_i \in \{-1, 1\}$ such that*

$$[K] = \sum_i \epsilon_i [K_i].$$

Therefore, we have that

$$f(K, \mathbf{x}) = \sum_i \epsilon_i f(K_i, \mathbf{x}).$$

Example 2..5 *Let $K = co\big((1,0),(1,N)\big)$ (that is, the cone generated by $(1,0)$ and $(1,N)$). Then $K_1 = co\big((1,0),(0,1)\big)$, $K_2 = co\big((1,N),(0,1)\big)$, and $K_3 = co\big((1,N)\big)$ are unimodular, with $[K] = [K_1] - [K_2] + [K_3]$. Therefore*

$$f(K; x, y)$$
$$= \tfrac{1}{(1-x)(1-y)} - \tfrac{1}{(1-xy^N)(1-y)} + \tfrac{1}{1-xy^N}.$$

Using Theorem 2..2, triangulation, Theorem 2..4, and (1), we may now compute the generating function for any rational polyhedron, as the following theorem states.

Theorem 2..6 (Theorem 4.4 in [6]) *Assume d, the dimension, is fixed. Given a rational polyhedron $P \subset \mathbb{R}^d$, the generating function $f(P;\mathbf{x})$ can be computed in polynomial time in the form*

$$f(P;\mathbf{x}) = \sum_{i\in I} \epsilon_i \frac{\mathbf{x}^{u_i}}{\prod_{j=1}^{d}(1-\mathbf{x}^{v_{ij}})}, \qquad (2)$$

where I is a polynomial-size indexing set, and where $\epsilon_i \in \{1,-1\}$ and $u_i, v_{ij} \in \mathbb{Z}^d$ for all i and j.

Furthermore, we can use this generating function to count the number of integer points in a rational polytope, by computing $f(P;1,1,\ldots,1)$. Note that, while $f(P;\mathbf{x})$ itself is analytic at $(1,1,\ldots,1)$, each of the fractions in the sum has a pole there. Thus, we can not directly substitute 1 for each variable, but we can compute the value (in polynomial time for fixed dimension) via residue calculus, as in [5].

3. Applications to integer programming

Throughout this section, we will assume that the number of variables d is fixed. Suppose we have the following integer programming problem: given $A \in \mathbb{Z}^{m\times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$,

$$\text{maximize } c\cdot x \ \text{ subject to } Ax \le b,\ x \in \mathbb{Z}^d. \qquad (3)$$

There are several algorithms to solve the integer programming problem (3) using short rational functions [12, 13, 14]. In this section, we will provide outlines of two of these algorithms: the *BBS algorithm* and the *digging algorithm*. These are currently implemented in the second release of the computer software `LattE` (see [11, 12, 15]).

3.1 Barvinok's binary search algorithm

We start with the most straightforward integer programming algorithm. It is an immediate consequence of the counting algorithm via short rational functions, with no extra tools needed: using binary search, one can turn *any* feasibility or counting oracle into an algorithm that solves Problem (3). This idea was proposed in [6]:

Algorithm 3..1 *(BBS algorithm)*
Input: $A \in \mathbb{Z}^{m\times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$.
Output: *The optimal value, O, of* $\max\{c\cdot x : Ax \le b,\ x \in \mathbb{Z}^d\}$.

```
1. Let $P = \{x \in \mathbb{R}^d : Ax \le b\}$ be the feasible region for the linear program. Initialize $M$ to be $\max\{c\cdot x : x \in P\}$, the maximum of the linear program, and $m$ to be $\min\{c\cdot x : x \in P\}$. Then $m \le O \le M$ (assuming a feasible integer solution exists).
2. Use Barvinok's algorithm to count $q = |P \cap \mathbb{Z}^d|$. If $q = 0$, then there is no feasible integer solution. If $q$ is nonzero, then perform the following loop.
3. While $M > m$ do (at each iteration, we will know that $m \le O \le M$)
     Set $N = \lceil \frac{M+m}{2} \rceil$.
     Using Barvinok's algorithm compute $q = |P \cap \{x \in \mathbb{R}^d : N \le c\cdot x \le M\} \cap \mathbb{Z}^d|$.
     If $q > 0$, then $N \le O \le M$ and we repeat the loop with $m := N$ and $M := M$.
     If $q = 0$, then $m \le O < N$ and we repeat the loop with $m := m$ and $M := N - 1$.
4. Return $M$ as the optimal value.
```

3.2 Digging algorithm

An alternative integer programming algorithm, which also uses rational generating functions, is the *digging algorithm*. This algorithm is an extension of a heuristic proposed by Lasserre [18], and it begins at the highest possible value for the optimum of the integer program, checks to see if there is a feasible solution giving that value, and if not continually *digs* to check the next highest possible value.

For the integer programming problem (3), let $P = \{x \in \mathbb{R}^d : Ax \le b\}$ be the feasible region for the *linear* program. Given the generating function $f(P;\mathbf{x})$

as in (2), perform the substitution $x_k = t^{c_k}$, so that

$$f(P; t^{c_1}, \cdots; t^{c_d}) = \sum_{\alpha \in P \cap \mathbb{Z}^d} t^{c \cdot \alpha} = \sum_{i \in I} \epsilon_i \frac{t^{c \cdot u_i}}{\prod_{j=1}^{d} (1 - t^{c \cdot v_{ij}})}. \tag{4}$$

For simplicity, we assume that $c \cdot v_{ij} \neq 0$, for any i, j. Furthermore, we may assume that $c \cdot v_{ij} < 0$ by performing the operation $1/(1-t^v) = -t^{-v}/(1-t^{-v})$, as necessary.

The optimum value of the integer program is the degree of $f(P; t^{c_1}, \cdots, t^{c_d})$, which we will compute by examining the Laurent series expressions of each of the terms in the sum, which are of the form

$$\epsilon_i t^{c \cdot u_i} \prod_{j=1}^{d} (1 + t^{c \cdot v_{ij}} + t^{2c \cdot v_{ij}} + t^{3c \cdot v_{ij}} + \cdots). \tag{5}$$

Algorithm 3..2 *(Digging algorithm)*
Input: $A \in \mathbb{Z}^{m \times d}$, $b \in \mathbb{Z}^m$, $c \in \mathbb{Z}^d$.
Output: *The optimal value of* $\max\{c \cdot x : Ax \leq b, \ x \in \mathbb{Z}^d\}$.

1. *Let $P = \{x \in \mathbb{R}^d : \ Ax \leq b\}$. Using Theorem 2..6 and substitution, compute the rational function expression in the equation (4). Apply the appropriate algebraic identities so that $c \cdot v_{ij} < 0$ for all i, j.*

2. *If $M = \max_i c \cdot u_i$, then the degree of $f(P; t^{c_1}, \cdots, t^{c_d})$ is at most M (since all of the $c \cdot v_{ij}$ are negative). Use the formula (5) to compute the coefficient of t^M in $f(P; t^{c_1}, \cdots, t^{c_d})$.*

3. *If the coefficient of t^M is nonzero, then M is the optimal value of the integer program and we are done. Otherwise, compute the coefficient of the next highest possible degree. Continue until a nonzero coefficient is found.*

Example 3..3 *Suppose we have the integer programming problem*

$$\begin{array}{l} \textit{maximize} \ \ 100x + 90y \\ \textit{subject to} \ x + y \leq 100, \ x \leq 50, \ x, y \geq 0, \ \ x, y \in \mathbb{Z}^d. \end{array}$$

Then the associated rational generating function for the feasible region of the integer program is

$$\frac{1}{(1-x_1)(1-x_2)} + \frac{x_1^{50}}{(1-x_1^{-1})(1-x_2)} + \frac{x_2^{100}}{(1-x_2^{-1})(1-x_1 x_2^{-1})} + \frac{x_1^{50} x_2^{50}}{(1-x_2^{-1})(1-x_1^{-1} x_2)}.$$

We apply the monomial substitution in (4) *and apply the appropriate algebraic identities, and we get:*

$$\frac{t^{-190}}{(1-t^{-100})(1-t^{-90})} - \frac{t^{4910}}{(1-t^{-100})(1-t^{-90})}$$
$$\frac{t^{8990}}{(1-t^{-90})(1-t^{-10})} + \frac{t^{9500}}{(1-t^{-90})(1-t^{-10})}.$$

Then, using Equation (5) *and the fact that the coefficient of t^{9500} is nonzero, we find that the integer programming optimum is* 9500.

3.3 Comparison of BBS and digging algorithms

How do these two algorithms compare? The Barvinok binary search algorithm is guaranteed to run in polynomial time if we fix d. However, it is usually slow in practice, because the Barvinok counting algorithm must be run a number of times. The digging algorithm, on the other hand, is not guaranteed to run in polynomial time with fixed d. Nevertheless, it is often quite efficient in practice, because the number of iterations in Step 3 of Algorithm 3..2 is usually small and Barvinok's algorithm is called only once (even if we vary cost functions, we do not have to re-run Barvinok's algorithm). This algorithm works well for small dimensions and polytopes with a small number of vertices. For example, the digging algorithm solved some hard knapsack problems which the mixed-integer programming solver `CPLEX` version 6.6. could not [13]. However, for d on the order of 30, this algorithm becomes quite slow. If the input polytope has a large number of vertices, the integer programming problem (3) can be solved by the following: First we find a vertex of the polytope which is an optimal solution for the linear relaxation of the problem (3) via linear programming. Then, we apply the digging calculation to the single tangent cone of the polytope at the vertex (see details in [13]).

REFERENCES

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

[2] K. Aardal, R. Weismantel, and L. A. Wolsey, *Non-standard approaches to integer programming*, Discrete Appl. Math., 123 (2002), pp. 5–74.

[3] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra, *Solving a linear diophantine equations with lower and upper bounds on the variables*, in *Integer Programming and Combinatorial Optimization, 6th International IPCO conference*, R. E. Bixby, E. A Boyd, and R. Z. Rios-Mercado, editors, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1412 (1998), pp. 229–242.

[4] F. Aurenhammer and R. Klein, *Voronoi diagrams*, in Handbook of Computational Geometry, J.-R. Sack and J. Urrutia, editors, North-Holland, Amsterdam, (2000) pp. 201–290.

[5] A. Barvinok, *Polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math. Oper. Res., 19 (1994) pp. 769–779.

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

[7] A. Barvinok and K. Woods, *Short rational generating functions for lattice point problems*, J. Amer. Math. Soc., 16 (2003), pp. 957–979.

[8] M. Brion, *Points entiers dans les polyèdres convexes*, Ann. Sci. École Norm. Sup., 21 (1988), pp. 653–663.

[9] W. Cook, T. Rutherford, H. E. Scarf, and D. Shallcross, *An implementation of the generalized basis reduction algorithm for integer programming*, ORSA Journal of Computing, 5 (1993), pp. 206–212.

[10] G. Cornuéjols, R. Urbaniak, R. Weismantel, and L. A. Wolsey, *Decomposition of integer programs and of generating sets*, in *Algorithms – ESA 97*, R. E. Burkard and G. J. Woeginger, editors, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1284 (1997), pp. 92–103.

[11] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida, *Effective lattice point counting in rational convex polytopes*, J. Symbolic Comput., 38 (2004), pp. 1273–1302.

[12] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, B. Sturmfels, and R. Yoshida, *Short rational functions for toric algebra and applications*, J. Symbolic Comput., 38 (2004), pp. 959–973.

[13] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, and R. Yoshida, *A computational study of integer programming algorithms based on Barvinok's rational functions*, Discrete Optimization, to appear.

[14] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, and R. Yoshida, *Three kinds of integer programming algorithms based on Barvinok's rational functions*, in *Integer Programming and Combinatorial Optimization: 10th International IPCO Conference*, D. Bienstock and G. Nemhauser, editors, Springer-Verlag, Berlin, (2004) pp. 244–255.

[15] J. A. De Loera, D. Haws, R. Hemmecke, P. Huggins, J. Tauzer, and Yoshida, *A user's guide for* `LattE` *v1.1.* 2003, software package. `LattE` is available at `http://www.math.ucdavis.edu/~latte`.

[16] M. Dyer and R. Kannan, *On Barvinok's algorithm for counting lattice points in fixed dimension*, Math. Oper. Res., 22 (1997), pp. 545–549.

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

[18] J. B. Lasserre, *Integer programming, Barvinok's counting algorithm and Gomory relaxations*, Oper. Res. Lett., 32 (2003), pp. 133–137.

[19] C. W. Lee, *Subdivisions and triangulations of polytopes*, in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, editors, CRC Press, New York, (1997) pp. 271–290.

[20] H. W. Lenstra, *Integer programming with a fixed number of variables*, Math. Oper. Res., 8 (1983), pp. 538–548.

[21] L. Lovász and H. E. Scarf, *The generalized basis reduction algorithm*, Math. Oper. Res., 17 (1992), pp. 751–764.

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

[23] R. Thomas, *Algebraic methods in integer programming*, in *Encyclopedia of Optimization*, C. Floudas and P. Pardalos, editors, Kluwer Academic Publishers, Dordrecht, 2001.

Lattice Basis Reduction in Integer Linear Optimization: Some Basic Topics

Karen Aardal[1]
Centrum voor Wiskunde en Informatica,
Amsterdam and Technische Universiteit Eindhoven,
The Netherlands (`Karen.Aardal@cwi.nl`).

1. Introduction and motivation

In this note I consider the following problems. First, the *integer linear feasibility problem.* Let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Ax} \le \boldsymbol{d}\}$, where the $m \times n$ matrix $\boldsymbol{A}$ and the m-vector $\boldsymbol{d}$ are given by integer input. Assume P is bounded and full-dimensional.

$$\text{Does there exist a vector } \boldsymbol{x} \in P \cap \mathbb{Z}^n? \qquad (1)$$

Second, the *integer linear optimization problem*:

$$\text{Determine a vector } \boldsymbol{x} \in P \cap \mathbb{Z}^n \text{ that maximizes } \boldsymbol{cx}, \qquad (2)$$

where $\boldsymbol{c}$ is an n-dimensional row vector. Branch-and-bound has proven to be a very successful method to solve integer linear optimization problems, especially when combined with cutting planes, and it is implemented in several commercial optimization packages. The key component of branch-and-bound is to solve the linear relaxation, i.e., we maximize (or minimize) $\boldsymbol{cx}$ over P. If the optimal solution $\bar{\boldsymbol{x}}^*$ to the linear relaxation is integer, then we are done, otherwise we choose a non-integral component of $\bar{\boldsymbol{x}}^*$, say x_j with value f_j and create two subproblems by adding the constraints $x_j \le \lfloor f_j \rfloor$, and $x_j \ge \lceil f_j \rceil$ respectively to P. This is the part of the algorithm that is called "branching". Each of the subproblems are again treated in the same fashion. We do not continue branching at a subproblem if the linear relaxation corresponding to that subproblem is infeasible, or if the optimal solution to the relaxation is integer, or if the value $\boldsymbol{c}\bar{\boldsymbol{x}}^*$ is "worse" than the value of the best known integer solution (this is the "bounding" part of the algorithm).

Sometimes, though, branch-and-bound does not work well. In some of these cases it seems as if the linear relaxation does not provide useful information, and in other cases we simply do not understand why it does not work. From the theoretical point of view we know that branch-and-bound can perform arbitrarily bad even in dimension $n = 2$. A natural question to ask is whether there exists an algorithm for solving Problem 1 that has a polynomial running time. Since Problem 1 is NP-complete [6, 20] we do not expect to find a polynomial algorithm to solve it, but we could still hope for an algorithm that is polynomial for fixed dimension n. Such an algorithm was developed by H. W. Lenstra, Jr. [23, 24]. His algorithm is based on lattice basis reduction and linear transformations, and is "branching on hyperplanes" rather than on information from the linear relaxation. Before describing Lenstra's algorithm in Section 4., we present some notation, definitions and basic results in Section 2.. In Section 3. we present Lovász' basis reduction algorithm, which is a corner stone in Lenstra's algorithm. Finally, in Sections 5. and 6. we present some related results and list some open problems.

2. Notation, definitions, and basic results

Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_l$ be linearly independent vectors in $\mathbb{R}^n$. The set

$$L = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} = \sum_{j=1}^{l} \lambda_j \boldsymbol{b}_j,\ \lambda_j \in \mathbb{Z},\ 1 \le j \le l\} \qquad (3)$$

is called a *lattice.* The set of vectors $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_l\}$ is called a *lattice basis.*

We review the Gram-Schmidt process below.

Definition 2..1 *The Gram-Schmidt process derives orthogonal vectors $\boldsymbol{b}_j^*$, $1 \le j \le l$, from linearly independent vectors $\boldsymbol{b}_j$, $1 \le j \le l$. The vectors $\boldsymbol{b}_j^*$, $1 \le j \le l$, and the real numbers μ_{jk}, $1 \le k < j \le l$, are determined from $\boldsymbol{b}_j$, $1 \le j \le l$, by the recursion*

$$\boldsymbol{b}_1^* = \boldsymbol{b}_1, \qquad \boldsymbol{b}_j^* = \boldsymbol{b}_j - \sum_{k=1}^{j-1} \mu_{jk} \boldsymbol{b}_k^*, \quad 2 \le j \le l,$$

where

$$\mu_{jk} = \frac{\boldsymbol{b}_j^T \boldsymbol{b}_k^*}{\|\boldsymbol{b}_k^*\|^2}, \quad 1 \le k < j \le l.$$

[1]The author was supported in part by NSF Grants DMI-0100020 and DMII-0121495, and the Dutch BSIK/BRICKS project.

The vector $\boldsymbol{b}_j^*$ is the projection of $\boldsymbol{b}_j$ on the orthogonal complement of $\sum_{k=1}^{j-1} \mathbb{R}\boldsymbol{b}_k = \{\sum_{k=1}^{j-1} m_k\boldsymbol{b}_k : m_k \in \mathbb{R},\ 1 \leq k \leq j-1\}$, i.e., $\boldsymbol{b}_j^*$ is the component of $\boldsymbol{b}_j$ orthogonal to the real subspace spanned by $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{j-1}$. Thus, any pair $\boldsymbol{b}_i^*$, $\boldsymbol{b}_k^*$ of the Gram-Schmidt vectors are mutually orthogonal. The multiplier μ_{jk} is equal to zero if and only if $\boldsymbol{b}_j$ is orthogonal to $\boldsymbol{b}_k^*$. Notice that the Gram-Schmidt vectors corresponding to $\boldsymbol{b}_1, \dots, \boldsymbol{b}_l$ do not necessarily belong to L, but they do span the same real vector space as $\boldsymbol{b}_1, \dots, \boldsymbol{b}_l$.

The *rank* of L, rk L, is equal to the dimension of the Euclidean vector space generated by the basis of L. The rank of the lattice L in Expression (3) is l, and we have $l \leq n$. If $l = n$ we call the lattice *full-dimensional*.

There are several ways of defining the *determinant of a given lattice* L, $d(L)$. Two straightforward expressions are:

$$d(L) = \|\boldsymbol{b}_1^*\| \cdot \|\boldsymbol{b}_2^*\| \cdot \ \cdots \ \cdot \|\boldsymbol{b}_l^*\|,$$

and

$$d(L) = \sqrt{\det(\boldsymbol{B}^T\boldsymbol{B})}\ ,$$

where $\boldsymbol{b}_1^*, \dots, \boldsymbol{b}_l^*$ are the Gram-Schmidt vectors corresponding to the lattice basis vectors $\boldsymbol{b}_1, \dots, \boldsymbol{b}_l$, and where $\boldsymbol{B}$ is the basis matrix $\boldsymbol{B} = (\boldsymbol{b}_1, \dots, \boldsymbol{b}_l)$. If L is full-dimensional, then $d(L)$ can be interpreted as the volume of the parallelepiped $\sum_{j=1}^n [0,1)\boldsymbol{b}_j$. In this case the determinant of the lattice can be computed straightforwardly as $d(L) = |\det(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)|$. The determinant of $\mathbb{Z}^n$ is equal to one. The rank and the determinant of L depend only on the lattice and not on the chosen basis (cf. Observation 3..3).

For more details about lattices, see e.g. Cassels [7], Grötschel, Lovász, and Schrijver [16], Lenstra [25], and Schrijver [31].

3. Lovász' basis reduction algorithm

In the lattice approaches to integer programming that will be discussed in Sections 4. and 5., we need lattice representations using bases with short, almost orthogonal basis vectors. Such bases are called reduced. Here we describe Lovász' basis reduction algorithm [22]. We also discuss some recent implementations.

Before discussing specific basis reduction algorithms we describe the basic operations that are used to go from one lattice basis to another.

Definition 3..1 *An integer nonsingular matrix* $\boldsymbol{U}$ *is* unimodular *if* $\det(\boldsymbol{U}) = \pm 1$.

The following operations on a matrix are called *elementary column operations*:

- exchanging two columns,
- multiplying a column by -1,
- adding an integer multiple of one column to another column.

Theorem 3..2 *An integer matrix* $\boldsymbol{U}$ *is unimodular if and only if* $\boldsymbol{U}$ *can be derived from the identity matrix by elementary column operations.*

Observation 3..3 *If* $\boldsymbol{B}$ *and* $\boldsymbol{B}'$ *are bases for the same lattice* $L \subset \mathbb{R}^n$ *of rank* l*, then* $\boldsymbol{B}' = \boldsymbol{B}\boldsymbol{U}$ *for some* $l \times l$ *unimodular matrix* $\boldsymbol{U}$.

A consequence of Theorem 3..2 and Observation 3..3 is that a lattice of rank greater than 1 has infinitely many bases. From Observation 3..3 we also see that the determinant of a lattice L only depends on L and not on the chosen basis.

To go from one basis to another is conceptually easy; given a basis $\boldsymbol{B}$ we just multiply $\boldsymbol{B}$ by a unimodular matrix, or equivalently, we perform a series of elementary column operations on $\boldsymbol{B}$, to obtain a new basis. The key question is of course how to do this efficiently so that the new basis is reduced according to the definition of reducedness we are using.

In Lovász' [22] basis reduction algorithm the length of the vectors are measured using the Euclidean length, and the Gram-Schmidt vectors corresponding to the current basis are used as a reference for checking whether the basis vectors are nearly orthogonal. Let $L \subset \mathbb{R}^n$ be a lattice, and let $\boldsymbol{b}_1, \dots, \boldsymbol{b}_l$, $l \leq n$, be the current basis vectors for L.

Definition 3..4 ([22]) *A basis* $\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_l$ *is called* reduced *if*

$$|\mu_{jk}| \leq \frac{1}{2} \quad \text{for } 1 \leq k < j \leq l, \tag{4}$$

$$\|\boldsymbol{b}_j^* + \mu_{j,j-1}\boldsymbol{b}_{j-1}^*\|^2 \geq \frac{3}{4}\|\boldsymbol{b}_{j-1}^*\|^2 \quad \text{for } 1 < j \leq l. \tag{5}$$

The constant $\frac{3}{4}$ in inequality (5) is arbitrarily chosen and can be replaced by any fixed real number $\frac{1}{4} < y < 1$. In a practical implementation one chooses a constant close to one.

Condition (4) is satisfied in two cases. The first, desired, case is if $\boldsymbol{b}_j$ is almost orthogonal to $\boldsymbol{b}_k^*$. The second possibility for (4) to be satisfied is if $\boldsymbol{b}_j$ is short relative to $\boldsymbol{b}_k^*$. If we would accept this case we would also accept a basis in which $\|\boldsymbol{b}_1\| >> \|\boldsymbol{b}_2\| >> \cdots >> \|\boldsymbol{b}_l\|$, and where the vectors are far from being orthogonal. To prevent this, Condition (5) is enforced. Here we relate to the interpretation of the Gram-Schmidt vectors above, and notice that the vectors $\boldsymbol{b}_j^* + \mu_{j,j-1}\boldsymbol{b}_{j-1}^*$ and $\boldsymbol{b}_{j-1}^*$ are the projections of $\boldsymbol{b}_j$ and $\boldsymbol{b}_{j-1}$ on the orthogonal complement of $\sum_{k=1}^{j-2} \mathbb{R}\boldsymbol{b}_k$. Consider a violation of Condition (4) where $k = j-1$, i.e., suppose that $\boldsymbol{b}_j$ is short compared to $\boldsymbol{b}_{j-1}^*$, which implies that $\boldsymbol{b}_j^*$ is short compared to $\boldsymbol{b}_{j-1}^*$ as $\|\boldsymbol{b}_j^*\| \leq \|\boldsymbol{b}_j\|$. Suppose we *interchange* $\boldsymbol{b}_j$ and $\boldsymbol{b}_{j-1}$. Then the new $\boldsymbol{b}_{j-1}^*$ will be the old $\boldsymbol{b}_j^* + \mu_{j,j-1}\boldsymbol{b}_{j-1}^*$, which will be short compared to the old $\boldsymbol{b}_{j-1}^*$, i.e., Condition (5) will be violated.

If Condition (4) is violated, then Lovász' basis reduction algorithm will replace $\boldsymbol{b}_j$ by $\boldsymbol{b}_j - \lceil\mu_{jk}\rfloor\boldsymbol{b}_k$, where $\lceil\mu_{jk}\rfloor := \lceil\mu_{jk} - \frac{1}{2}\rceil$. Such a step is called *size reduction* and will ensure nearly orthogonal basis vectors. If Condition (5) is violated for a certain index j, then the vectors $\boldsymbol{b}_j$ and $\boldsymbol{b}_{j-1}$ are *interchanged*, which ensures that the vectors are ordered, up to a multiplicative constant, in increasing order of their Euclidean length. It is clear that both the size reduction and the interchange are elementary column operations. For details on the precise sequence of the size reductions and interchanges, we refer to [22].

Theorem 3..5 ([22]) *Let $L \subseteq \mathbb{Z}^n$ be a lattice with basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, and let $\beta \in \mathbb{R}$, $\beta \geq 2$, be such that $\|\boldsymbol{b}_j\|^2 \leq \beta$ for $1 \leq j \leq n$. Then the number of arithmetic operations needed by the basis reduction algorithm as described in [22] is $O(n^4 \log\beta)$, and the integers on which these operations are performed each have binary length $O(n\log\beta)$.*

In terms of bit operations, Theorem 3..5 implies that Lovász' basis reduction algorithm has a running time of $O(n^6(\log\beta)^3)$ using classical algorithms for addition and multiplication.

Next we will present a few useful bounds on reduced basis vectors. In Proposition 3..6 and Corollary 3..7 we assume for simplicity that the lattice L is full-dimensional.

Proposition 3..6 ([22]) *Let $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ be a reduced basis for the lattice $L \subset \mathbb{R}^n$. Then,*

$$\|\boldsymbol{b}_1\|^2 \leq 2^{n-1}\|\boldsymbol{x}\|^2 \quad \textit{for all } \boldsymbol{x} \in L,\ \boldsymbol{x} \neq 0, \tag{6}$$

and

$$d(L) \leq \Pi_{j=1}^n \|\boldsymbol{b}_j\| \leq c_1 \cdot d(L), \tag{7}$$

where $c_1 = 2^{n(n-1)/4}$.

Expression (6) implies that the first basis vector in a reduced basis is an approximation of the shortest non-zero lattice vector. The first inequality in (7) is also referred to as the *inequality of Hadamard* that holds for any basis of L. Hadamard's inequality holds with equality if and only if the basis is orthogonal. Hermite [17] proved that each lattice $L \subset \mathbb{R}^n$ has a basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ such that $\Pi_{j=1}^n \|\boldsymbol{b}_j\| \leq c \cdot d(L)$, where c is a constant depending only on n. The basis produced by Lovász' basis reduction algorithm yields the constant $c = c_1$ in Proposition 3..6. Better constants than c_1 are possible, but the question is then whether the basis can be obtained in polynomial time.

A consequence of Proposition 3..6 is that if we consider a basis that satisfies (7), then the distance of the basis vector $\boldsymbol{b}_n$ to the hyperplane generated by the reduced basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_{n-1}$ is not too small as stated in the following corollary.

Corollary 3..7 ([24]) *Assume that $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is a basis such that (7) holds, and that, after possible reordering, $\|\boldsymbol{b}_n\| = \max_{1\leq j\leq n}\{\|\boldsymbol{b}_j\|\}$. Let $H = \sum_{j=1}^{n-1} \mathbb{R}\boldsymbol{b}_j$ and let h be the distance of the basis vector $\boldsymbol{b}_n$ to H. Then*

$$c_1^{-1} \cdot \|\boldsymbol{b}_n\| \leq h \leq \|\boldsymbol{b}_n\|, \tag{8}$$

where $c_1 = 2^{n(n-1)/4}$.

The lower bound on h given in Corollary 3..7 plays a crucial role in Lenstra's algorithm that is described in the next section.

In recent years several new variants of Lovász' basis reduction algorithm have been developed, and a number of variants for implementation have been suggested. We recommend the paper by Schnorr and Euchner [30] and the book chapter by Aardal and

Eisenbrand [2] for a more detailed overview. For the reader interested in using a version of Lovász' basis reduction algorithm there are some useful libraries available on the Internet. In particular we want to mention NTL — a Library for doing Number Theory, developed by V. Shoup [33].

4. Lenstra's algorithm

As mentioned in the introduction, if one uses branch-and-bound for solving problem (1) it is possible, even in dimension 2, to create an arbitrarily deep search tree for certain thin polytopes. Lenstra [24] suggested transforming the polytope P using a linear transformation τ such that the polytope τP becomes "round" according to a certain measure. Assume, without loss of generality, that the polytope P is full-dimensional and bounded, and let $B(\boldsymbol{p}, z) = \{\boldsymbol{x} \in \mathbb{R}^n : \|\boldsymbol{x} - \boldsymbol{p}\| \le z\}$ be the closed ball with center $\boldsymbol{p}$ and radius z. The transformation τ that we apply to the polytope is constructed so that $B(\boldsymbol{p}, r) \subset \tau P \subset B(\boldsymbol{p}, R)$ for some $\boldsymbol{p} \in \tau P$, with r and R satisfying

$$\frac{R}{r} \le c_2, \tag{9}$$

where c_2 is a constant that depends only on the dimension n. Relation (9) is the measure of "roundness" that Lenstra uses. Once we have transformed the polytope, we need to apply the same transformation to the lattice, which gives us the following feasibility problem that is equivalent to problem (1):

$$\text{Is } \tau\mathbb{Z}^n \cap \tau P \ne \emptyset?$$

The vectors $\tau\boldsymbol{e}_j$, $1 \le j \le n$, where $\boldsymbol{e}_j$ is the j-th unit vector in $\mathbb{R}^n$, form a basis for the lattice $\tau\mathbb{Z}^n$. If the polytope P is thin, then this will translate to the lattice basis vectors $\tau\boldsymbol{e}_j$, $1 \le j \le n$, in the sense that these vectors are long and non-orthogonal. This is where lattice basis reduction becomes useful. Once we have the transformed polytope τP, Lenstra uses the following lemma to find a lattice point quickly.

Lemma 4..1 ([24]) *Let $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ be any basis for L. Then for all $\boldsymbol{x} \in \mathbb{R}^n$ there exists a vector $\boldsymbol{y} \in L$ such that*

$$\|\boldsymbol{x} - \boldsymbol{y}\|^2 \le \frac{1}{4}(\|\boldsymbol{b}_1\|^2 + \cdots + \|\boldsymbol{b}_n\|^2).$$

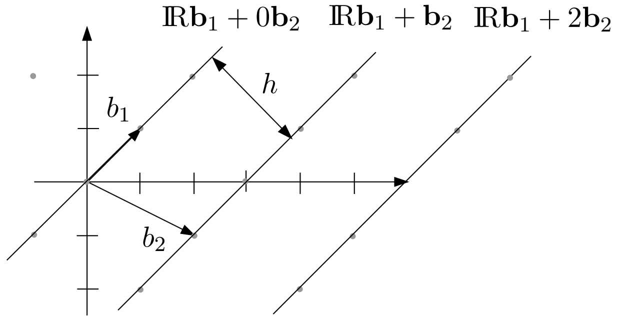
$\mathbb{R}\mathbf{b}_1 + 0\mathbf{b}_2$
$\mathbb{R}\mathbf{b}_1 + \mathbf{b}_2$
$\mathbb{R}\mathbf{b}_1 + 2\mathbf{b}_2$
h
b_1
b_2

Figure 1: The lattice L is contained in countably many parallel hyperplanes.

The proof of this lemma suggests a polynomial time construction of the vector $\boldsymbol{y} \in L$ given the vector $\boldsymbol{x}$.

Next, let $L = \tau\mathbb{Z}^n$, and let $\boldsymbol{b}_1, \dots, \boldsymbol{b}_n$ be a basis for L such that (7) holds. Notice that (7) holds if the basis is reduced. Also, reorder the vectors so that $\|\boldsymbol{b}_n\| = \max_{1 \le j \le n}\{\|\boldsymbol{b}_j\|\}$. After reordering, the basis might not be reduced any longer, but Condition (7) still holds. Let $\boldsymbol{x} = \boldsymbol{p}$ where $\boldsymbol{p}$ is the center of the closed balls $B(\boldsymbol{p}, r)$ and $B(\boldsymbol{p}, R)$. Apply Lemma 4..1 to the given $\boldsymbol{x}$. This gives a lattice vector $\boldsymbol{y} \in \tau\mathbb{Z}^n$ such that

$$\|\boldsymbol{p} - \boldsymbol{y}\|^2 \le \frac{1}{4}(\|\boldsymbol{b}_1\|^2 + \cdots + \|\boldsymbol{b}_n\|^2) \le \frac{1}{4} \cdot n \cdot \|\boldsymbol{b}_n\|^2 \tag{10}$$

in polynomial time. We now distinguish two cases. Either $\boldsymbol{y} \in \tau P$ or $\boldsymbol{y} \notin \tau P$. In the first case we are done, so assume we are in the second case. Since $\boldsymbol{y} \notin \tau P$ we know that $\boldsymbol{y}$ is not inside the ball $B(\boldsymbol{p}, r)$ as $B(\boldsymbol{p}, r)$ is completely contained in τP. Hence we know that $\|\boldsymbol{p} - \boldsymbol{y}\| > r$, or using (10), that

$$r < \frac{1}{2} \cdot \sqrt{n} \cdot \|\boldsymbol{b}_n\|. \tag{11}$$

Let H and h be defined as in Corollary 3..7 of Section 3., and let $L' = \sum_{j=1}^{n-1} \mathbb{Z}\boldsymbol{b}_j$. We can write L as

$$L = L' + \mathbb{Z}\boldsymbol{b}_n \subset H + \mathbb{Z}\boldsymbol{b}_n = \cup_{k \in \mathbb{Z}}(H + k\boldsymbol{b}_n). \tag{12}$$

So the lattice L is contained in countably many parallel hyperplanes. For an example we refer to Figure 1. The distance between two consecutive hyperplanes is h, and Corollary 3..7 says that h is bounded from below by $c_1^{-1}\|\boldsymbol{b}_n\|$, which implies that not too many hyperplanes intersect τP. To determine precisely how many hyperplanes intersect τP, we approximate τP by the ball $B(\boldsymbol{p}, R)$. If t is the number of hyperplanes intersecting $B(\boldsymbol{p}, R)$ we have

$$t - 1 \leq \frac{2R}{h}.$$

Using the relationship (9) between the radii R and r we have $2R \leq 2rc_2 < c_2\sqrt{n}\|\boldsymbol{b}_n\|$, where the last inequality follows from (11). Since $h \geq c_1^{-1}\|\boldsymbol{b}_n\|$, we get the following bound on the number of hyperplanes that we need to consider:

$$t - 1 \leq \frac{2R}{h} < c_1 c_2 \sqrt{n},$$

which depends on the dimension only. The values of the constants c_1 and c_2 that are used by Lenstra are: $c_1 = 2^{n(n-1)/4}$ and $c_2 = 2n^{3/2}$. Lenstra discusses ways of improving these values. To determine the values of k in Expression (12), we express $\boldsymbol{p}$ as a linear combination of the basis vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$. Recall that $\boldsymbol{p}$ is the center of the ball $B(\boldsymbol{p}, R)$ that was used to approximate τP.

Below we will describe the tree search algorithm and argue why it is polynomial for fixed n. Let us first consider the root node of the search tree. We create $t < c_1 c_2 \sqrt{n} + 1$ subproblems by considering intersection between τP with t parallel lattice hyperplanes. Each of the subproblems has dimension at least one lower than the parent problem. The procedure of splitting the problem into subproblems of lower dimension is called "branching", and each subproblem is represented by a node in the enumeration tree. In each node we repeat the whole process of transformation, basis reduction and, if necessary, branching. The enumeration tree created by this recursive process is of depth at most n, and the number of nodes at each level is bounded by a constant that depends only on the dimension.

For the details on how to determine the transformation τ and hence the balls $B(\boldsymbol{p}, r)$ and $B(\boldsymbol{p}, R)$ we refer to Lenstra [24] or the survey chapter by Aardal and Eisenbrand [2]. Lenstra's algorithm has been implemented by Gao and Zhang [14], and a heuristic version of the algorithm has been developed and implemented by Aardal et al. [1], and Aardal and Lenstra [4].

5. A few related results

In the working paper by Lenstra [23], he used a basis reduction algorithm that is polynomial for fixed n only. Soon after, Lovász developed the basis reduction algorithm described in Section 3.. This is the basis reduction algorithm used in the published paper by Lenstra [24], so even though the result in Theorem 5..1 below is present in Lenstra's paper, the result is credited to Lovász (see for instance Schrijver [31], pp 256–259). Notice that "polynomial time" in Theorem 5..1 is polynomial time for *varying* n.

Theorem 5..1 *Let $\boldsymbol{Ax} \leq \boldsymbol{b}$ be a system of m rational inequalities in n variables, and let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{Ax} \leq \boldsymbol{b}\}$. There exists a polynomial time algorithm that finds either an integer vector $\boldsymbol{y} \in P$, or a vector $\boldsymbol{d} \in \mathbb{Z}^n \setminus \{\boldsymbol{0}\}$ such that*

$$\begin{aligned}&\max\{\boldsymbol{dx} \mid \boldsymbol{x} \in P\} - \min\{\boldsymbol{dx} \mid \boldsymbol{x} \in P\}\\ \leq\ & 2n(n+1)2^{n(n-1)/4}\end{aligned}$$

Lenstra's result that Problem 1 can be solved in polynomial time for fixed number of variables can be obtained as a corollary to Theorem 5..1.

Kannan [19] developed a variant of Lenstra's algorithm. The algorithm follows Lenstra's algorithm up to the point where he has applied a linear transformation to the polytope P and obtained a polytope τP such that $B(\boldsymbol{p}, r) \subset \tau P \subset B(\boldsymbol{p}, R)$ for some $\boldsymbol{p} \in \tau P$. Here Kannan proceeds as follows. He applies a reduction algorithm to a basis of the lattice $\tau\mathbb{Z}^n$ that produces a "reduced" basis defined differently from a Lovász' reduced basis. In particular, in Kannan's reduced basis the first basis vector is the shortest nonzero lattice vector. As in Lenstra's algorithm two cases are considered. Either τP is relatively large, which implies that τP contains a lattice vector, or τP is small, which means that not too many lattice hyperplanes can intersect τP. Each such intersection gives rise to a subproblem of at least one dimension lower. Kannan's reduced basis makes it possible to improve the bound on the number of hyperplanes that has to be considered to $O(n^{5/2})$.

Lovász and Scarf [27] developed a basis reduction algorithm called "generalized basis reduction". Here, the norm used is related to the polytope P.

Based on their basis reduction algorithm, and several other results, they suggested an integer programming algorithm that also uses the "branching on hyperplanes" idea, where we branch in "thin" directions. A heuristic version of Lovász and Scarf's integer programming algorithm has been implemented by Cook et al. [9].

Barvinok [5] generalized Lenstra's result by giving an algorithm that counts integer points in a polytope in polynomial time if the dimension is fixed. Later, Dyer and Kannan [12] developed a simplification of Barvinok's algorithm. Barvinok's algorithm has been implemented by De Loera et al. [11] who also introduced further practical improvements. In a different chapter of this newsletter Yoshida and Woods present Barvinok's algorithm and related results.

Cook, Hartmann, Kannan, and McDiarmid [8] proved that if P_I is the integer hull of the rational polyhedron $P \subset \mathbb{R}^n$ given by m inequalities whose size is at most φ, then for fixed n an upper bound on the number of vertices of P_I is $\mathrm{O}(m^n \varphi^{n-1})$. This implies that integer hull computations can be done in polynomial time for fixed n as well.

The integer optimization problem (2) can be reduced to the integer feasibility problem by applying binary search. The running time of such an algorithm is $\mathrm{O}(m\varphi + \varphi^2)$, where φ is the maximum size of any of the constraints and the objective vector $\boldsymbol{c}$. Eisenbrand [13] developed an algorithm for the integer optimization problem with a fixed number, m, of constraints that uses $\mathrm{O}(\varphi)$ arithmetic operations on rational numbers of binary encoding length $\mathrm{O}(\varphi)$. By combining his results with a randomized algorithm by Clarkson [10], Eisenbrand obtains an $\mathrm{O}(m + \log m\, \varphi)$ algorithm for the integer optimization problem.

One of the first computational successes of basis reduction was its use to break knapsack cryptosystems, see Lagarias and Odlyzko [21]. Inspired by this work Aardal, Hurkens, and Lenstra [3] suggested a reformulation based on lattice bases. This reformulation has been successful when solving some difficult instances of integer feasibility problems such as the market split problem [1] and the integer knapsack problem [4]. Louveaux and Wolsey [26] have used a similar lattice reformulation to solve a portfolio planning problem.

6. Some open problems and further reading

Computationally there is a lot to be done with respect to branching on hyperplane algorithms. The few implementations that exist (see e.g. [1, 4, 9, 14]) all seem to agree that fewer, in many cases significantly fewer, search nodes are needed compared to branch-and-bound. Each node, however, requires more work as a good search direction has to be computed instead of just solving a linear programming relaxation. More work is needed on determining "practically good" search directions, rather than just more or less following the theoretical proofs of the various algorithms, in order to profit from the gain in the reduced size of the search trees. Another bottleneck is the basis reduction algorithm. While polynomial, it still becomes slow when the problem size exceeds a couple of hundred variables. Would it be possible to develop a faster basis reduction algorithm with no worse worst case bounds than the ones given in Expression 6? Goldreich and Goldwasser [15] have shown that approximating the shortest nonzero vector in a given lattice with an approximation bound of $\mathrm{O}(\sqrt{n})$ is not NP-hard unless the polynomial hierarchy collapses. It should be pointed out though that, in practice, Lovász' basis reduction algorithm produces basis vectors that are much better than the theoretical bounds suggest. This is one of the reasons why the algorithm has become such a useful tool in many applications such as cryptography, see eg. Nguyen and Stern [29]. Why the algorithm performs so well is not well understood.

For further reading on this topic we refer to the overview article by Kannan [18], and the books by Grötschel, Lovász, and Schrijver, [16], and by Schrijver [31]. Branch-and-bound and cutting plane related integer programming algorithms are treated in depth in the book by Nemhauser and Wolsey [28]. Aardal and Eisenbrand [2] give an overview on lattice based integer programming as well as results on integer hulls and cutting plane closures in fixed dimension.

REFERENCES

[1] K. Aardal, R. E. Bixby, C. A. J. Hurkens, A. K. Lenstra, and J. W. Smeltink, *Market split and basis*

reduction: Towards a solution of the Cornuéjols-Dawande instances, INFORMS J. Comput., 12 (2000), pp. 192–202.

[2] K. Aardal and F. Eisenbrand, *Integer programming, lattices, and results in fixed dimension*, to appear as Chapter 4 in *Handbook on Discrete Optimization* in the series *Handbooks in Operations Research and Management Science*, Elsevier, Amsterdam.

[3] K. Aardal, C. A. J. Hurkens, and A. K. Lenstra, *Solving a system of diophantine equations with lower and upper bounds on the variables*, Math. Oper. Res., 25 (2000), pp. 427–442.

[4] K. Aardal and A. K. Lenstra, *Hard equality constrained integer knapsacks*, Math. Oper. Res., 29 (2004), pp. 724–738.

[5] A. I. Barvinok, *A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed*, Math. Oper. Res., 19 (1994), pp. 769–779.

[6] I. Borosh and L. B. Treybig, *Bounds on positive integral solutions of linear diophantine equations*, Proc. Amer. Math. Soc., 55 (1976), pp. 299–304.

[7] J. W. S. Cassels, *An Introduction to the Geometry of Numbers*, second printing (reprint of the 1971 edition), Springer-Verlag, Berlin, 1997.

[8] W. Cook, M. E. Hartmann, R. Kannan, and C. McDiarmid, *On integer points in polyhedra*, Combinatorica, 12 (1992), pp. 27–37.

[9] W. Cook, T. Rutherford, H. E. Scarf, and D. Shallcross, *An implementation of the generalized basis reduction algorithm for integer programming*, ORSA Journal on Computing, 5 (1993), pp. 206–212.

[10] K. L. Clarkson, *Las Vegas algorithms for linear and integer programming when the dimension is small*, J. ACM, 42 (1995), pp. 448–499.

[11] J. A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida, *Effective lattice point counting in rational polytopes*, J. Symbolic Comput., 38 (2004), pp. 1273–1302.

[12] M. E. Dyer and R. Kannan, *On Barvinok's algorithm for counting lattice points in fixed dimension*, Math. Oper. Res., 22 (1997), pp. 545–549.

[13] F. Eisenbrand, *Fast integer programming in fixed dimension*, in *Algorithms – ESA 2003*, G. Di Battista and U. Zwick, editors, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2832 (2003) pp. 196–207.

[14] L. Gao and Y. Zhang, *Computational experience with Lenstra's algorithm*, Technical report TR02-12, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2002.

[15] O. Goldreich and S. Goldwasser, On the limits of non-approximability of lattice problems, in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, (1998), pp. 1–9.

[16] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[17] C. Hermite, *Extraits de lettres de M. Ch. Hermite à M. Jacobi sur différents objets de la théorie des nombres*, J. Reine Angew. Math., 40 (1850).

[18] R. Kannan, *Algorithmic geometry of numbers*, Annual Review of Computer Science, 2 (1987), pp. 231–267.

[19] R. Kannan, *Minkowski's convex body theorem and integer programming*, Math. Oper. Res., 12 (1987), pp. 415–440.

[20] R. M. Karp, *Reducibility among combinatorial problems*, in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, editors, Plenum Press, New York, NY, (1972), pp. 85–103.

[21] J. C. Lagarias and A. M. Odlyzko, *Solving low-density subset sum problems*, J. ACM, 32 (1985), pp. 229–246.

[22] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Math. Ann., 261 (1982), pp. 515–534.

[23] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Report 81-03, Department of Mathematics, University of Amsterdam, 1981.

[24] H. W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Math. Oper. Res., 8 (1983), pp. 538–548.

[25] H. W. Lenstra, Jr., *Flags and lattice basis reduction*, in *Proceedings of the third European Congress of Mathematics Volume I*, C. Casacuberta, R. M. Miró-Roig, J. Verdera, and S. Xambó-Descamps, editors, Birkhäuser Verlag, Basel, (2000) pp. 37–51.

[26] Q. Louveaux and L. A. Wolsey, *Combining problem structure with basis reduction to solve a class of hard integer programs*, Math. Oper. Res., 27 (2002), pp. 470–484.

[27] L. Lovász and H. E. Scarf, *The generalized basis reduction algorithm*, Math. Oper. Res., 17 (1992), pp. 751–764.

[28] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

[29] P. Q. Nguyen and J. Stern, *The two faces of lattices in cryptology*, in *Cryptography and Lattices, International Conference, CaLC 2001*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 2146 (2001), pp. 146–180.

[30] C.-P. Schnorr and M. Euchner, *Lattice basis reduction: improved practical algorithms and solving subset sum problems*, Math. Program., 66 (1994), pp. 181–199.

[31] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, New York, 1986.

[32] M. Seysen, *Simultaneous reduction of a lattice basis and its reciprocal basis*, Combinatorica, 13 (1993), pp. 363–376.

[33] V. Shoup, *NTL: A Library for doing Number Theory*, Courant Institute, New York. http://www.shoup.net.

Article

Optimizing the Quality of Mesh Elements

Todd S. Munson
Mathematics and Computer Science Division,
Argonne National Laboratory,
Argonne, IL 60439, USA (tmunson@mcs.anl.gov).

1. Introduction

Discretization methods are common techniques for computing approximate solutions to partial differential equations [7, 10, 30]. These methods decompose the given domain into a finite set of elements, triangles or tetrahedrons, for example, to produce a mesh used within the approximation scheme. Several factors affect the accuracy of the solution obtained: the degree of the approximation scheme and the number of elements in the mesh [2], and the quality of the mesh [4, 5]. Optimizing the quality of the mesh prior to computing the approximate solution can improve the condition number of the linear systems solved [29], reduce the time taken to compute the solution [15], and increase the numerical accuracy.

The savings in computational time from using the optimized mesh can be substantial. One application we investigated was to solve the Navier-Stokes equations for a fluid with a moderate Reynolds number containing a dilute suspension of particles [34]. The approximate solution was obtained by applying a spectral element method to a hexahedral mesh. The top portion of Figure 1 shows the original mesh constructed by applying the meshing technique developed by Lin Zhang, while the bottom depicts the optimized mesh. The original mesh has many regular elements, while the optimized mesh loses much of this structure. However, the spectral element method applied required 29 hours to compute a solution when using the original mesh, but only 20 hours when using the optimized mesh, a 30% reduction in time. The optimization problem was modeled in AMPL [11] and solved by applying KNITRO

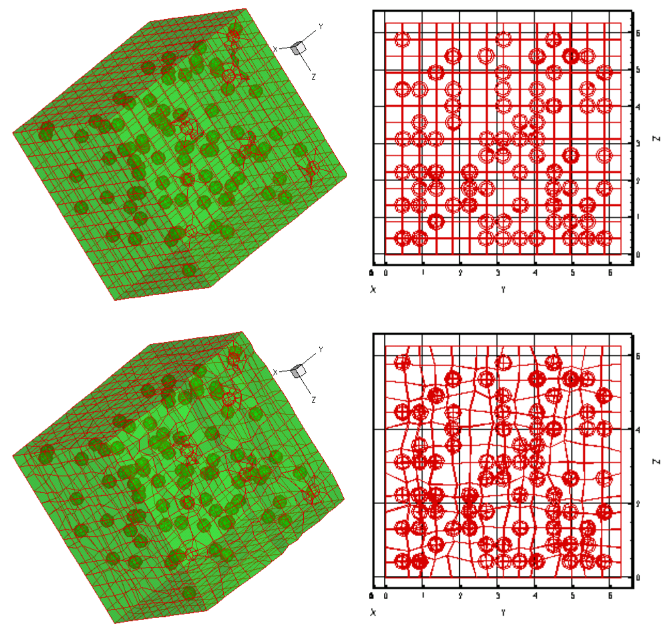

Figure 1: Original mesh (top left) and side view (top right) and optimized mesh (bottom left) and side view (bottom right) for fluid dynamics example.

[8, 33]. This instance had 18,135 variables, 1,170 linear constraints, and 3,004 nonlinear constraints. Computing an optimal mesh took approximately 33 minutes on a 2.4 GHz Intel Xeon workstation with 2 GB RAM; AMPL consumed 265 MB RAM to generate the problem, while KNITRO allocated 597 MB RAM to solve it.

The optimization problem we solve computes positions for the vertices in a given mesh to improve the average element quality according to a metric [13]; we do not change the mesh topology. Many metrics have been applied to such problems [3, 12, 14, 19, 20, 27]. We use the inverse mean-ratio metric [21, 20], a shape-quality metric measuring the distance between a trial element and an ideal element, a regular tetrahedron, for example. The objective function for the resulting optimization problem is nonconvex and consists of the sum of many fractional terms. The optimization problem and its properties are developed in Section 2. Proofs for the claims made in this section are found in [22, 23].

Applications with moving meshes or deforming geometries may require a mesh optimization step every time the domain is modified. For example, the particles in the fluid dynamics application could move as a function of time. In general, the time required to optimize the mesh must not dominate the computational savings achieved when solving the partial differential equation. Therefore, we want to compute an optimal mesh in a minimal amount of time and with minimal memory requirements. We discuss in Section 3 several techniques for improving the performance of the KNITRO libraries on an unconstrained version of the mesh optimization problem.

In Section 4 we discuss some of the lessons learned while working on this application. We also mention some of the complications associated with efficiently solving the fluid dynamics example where the vertices on the planar boundaries of the mesh are allowed to move within the plane.

2. Mesh optimization problem

The mesh optimization problem we solve minimizes the average inverse mean-ratio metric referenced to an ideal element. The description of this metric follows that of Knupp [19, 20] and Freitag and Knupp [12]. We discuss the metric only for tetrahedral elements. Each tetrahedron is defined by four vertices $[x_1, x_2, x_3, x_4]$, where each vertex belongs to $\mathbb{R}^3$. Other element types can be modeled by decomposing them into tetrahedrons. Hexahedral elements, for example, are decomposed into eight overlapping tetrahedral elements.

The development of the inverse-mean ratio metric begins by constructing the incidence function $A : \mathbb{R}^{3\times 4} \to \mathbb{R}^{3\times 3}$:

$$A(x) := [x_2 - x_1, x_3 - x_1, x_4 - x_1]\,.$$

This function computes a matrix containing the edges emanating from the first vertex of the element. The volume of the tetrahedron is related to the determinant of $A(x)$, which can be positive or negative depending on the labeling of the vertices. We assume throughout that the vertices are labeled according to the right-hand rule so that the determinant is nonnegative.

Two elements x and y have the same shape if their edges are proportional. That is,

$$A(x) = \sigma A(y)$$

for some $\sigma > 0$. In particular, if two elements with nonzero volume have the same shape, then

$$\|A(x)A(y)^{-1}\|_F^2 = \|\sigma I\|_F^2 = 3\sigma^2$$

and

$$\det(A(x)A(y)^{-1}) = \det(\sigma I) = \sigma^3.$$

The inverse mean-ratio referenced to the given element y is then defined by a function $Q_y : \mathbb{R}^{3\times 4} \to \mathbb{R}$ that takes the ratio of these two quantities. In particular,

$$Q_y(x) := \frac{\|A(x)A(y)^{-1}\|_F^2}{3\det(A(x)A(y)^{-1})^{2/3}}.$$

This metric has a value of one if x and y have the same shape and a value greater than one if their shapes differ. Moreover, it is translation, rotation, and scale invariant. The metric values are preserved when the same even permutation is applied to the columns of x and y. Since y is fixed, this metric is computed by using the QR factorization of $A(y)$ so that we multiply $A(x)$ only by the upper triangular matrix R^{-1}. This modification reduces the number of floating-point operations required to compute the function, gradient, and Hessian of Q_y.

A mesh consists of a set of vertices V and the elements E connecting these vertices, where each element is an ordered set of four indices. The optimization problem to minimize the average inverse mean-ratio metric is then

$$\begin{array}{ll} \displaystyle\min_{x\in\mathbb{R}^{3\times|V|}} & \theta(x) := \displaystyle\frac{1}{|E|}\sum_{e\in E}\frac{\|A(x_e)A(y)^{-1}\|_F^2}{3\det(A(x_e)A(y)^{-1})^{2/3}} \\ \text{subject to} & \det(A(x_e)A(y)^{-1}) > 0 \quad \forall e \in E \\ & x_i \in X_i \qquad\qquad\qquad\quad \forall i \in V, \end{array}$$

where x_e denotes the matrix of coordinates for element e and X_i is a set restricting the feasible locations for vertex i. In particular, the vertices on the boundary of the mesh are usually either fixed in space or constrained to lie on a particular piece of the boundary of the domain, while the other vertices are unrestricted. The volume constraints, the strict inequalities in the optimization problem, ensure a consistent orientation in the resulting mesh. A consistent orientation for all the elements is required for standard discretization methods to work correctly [7]. The objective function is twice continuously differentiable at all feasible points but is not necessarily convex on this region. Furthermore, the feasible region may be neither convex nor connected. While the Hessian of the objective function may not be positive definite, one can prove that $\nabla^2_{x_i,x_i}\theta(x)$ is positive definite for all i [22, 23].

The volume constraints are problematic because they involve a strict inequality. If at least two vertices are fixed in position and the mesh is edge connected (between any two elements there is a sequence of elements whose neighbors share a common edge), then the objective function approaches infinity for any sequence of feasible points in which the volume of at least one element approaches zero [22]. Therefore, the volume orientation constraints can be dropped to produce the optimization problem

$$\begin{array}{ll} \displaystyle\min_{x\in\mathbb{R}^{3\times|V|}} & \displaystyle\frac{1}{|E|}\sum_{e\in E}\frac{\|A(x_e)A(y)^{-1}\|_F^2}{3\max\left\{\det(A(x_e)A(y)^{-1}),0\right\}^{2/3}} \\ \text{subject to} & x_i \in X_i \quad \forall i \in V. \end{array}$$

In this case, the objective function is defined to have a value of plus infinity whenever the volume of at least one element is nonpositive. With this reformulation, we must provide a starting point where the orientation constraints are satisfied. However, most meshing packages used to construct the original mesh for the given domain, such as [26, 28], provide a set of vertices satisfying the orientation constraints.

3. Unconstrained results

In this section, we assume that all vertices on the boundary of the domain are fixed in position and the remaining vertices are unrestricted. Once the boundary vertices are removed from the problem, we are left with an unconstrained optimization problem with an objective function that is twice continuously differentiable on an open set containing the level set for the given feasible mesh.

The resulting unconstrained optimization problem was modeled in AMPL [11] and solved by applying KNITRO 4.0 [8, 33] and LOQO 6.06 [31, 32]. We always used the Interior/CG version of KNITRO and the default version of LOQO for these tests. The results on a representative set of test meshes are given in Table 1, where the number of variables and nonzeros in the Hessian matrix are provided for each example. The times are reported in seconds on a 2.0

Table 1: Unconstrained results using AMPL.

Mesh	Variables	Nonzeros	KNITRO	LOQO
gear	780	8,256	1.87	2.05
foam5	867	10,518	2.34	3.20
hook	1,200	16,872	3.15	5.84
duct20	1,146	17,601	3.12	14.06
duct15	2,895	50,106	7.96	30.80
duct12	6,906	129,102	21.59	108.44
duct10	13,440	262,329	49.27	160.96
duct8	26,214	529,212	124.81	929.33
duct4	425,952	9,209,799	-	-
duct2	3,323,229	45,882,111	-	-

GHz Intel Xeon workstation with 4 GB RAM. No other users were on the workstation when the results were generated, and all data and executables were on local disk drives. Each problem was run three times; the lowest time is reported. The largest models could not be solved because of memory requirements. In particular, AMPL consumed 567 MB to generate the duct8 instance, while KNITRO allocated 1,055 MB to solve it and LOQO used 1,008 MB.

Using AMPL provides many advantages: models can be quickly constructed, derivatives are automatically generated for the functions, and many numerical methods are readily available to solve the resulting instances. In particular, one can obtain results for the mesh optimization problems in a few days. However, a price is paid for this convenience: it can take considerable time to compute the optimal mesh, and the amount of memory consumed can be large.

Since our ultimate goal is to embed mesh optimization within a larger code for solving partial differential equations, we must compute a solution in a short amount of time with a small memory requirement. Therefore, we implemented a simple framework that reads the description of a mesh from a file, constructs the unconstrained optimization problem, calls an optimization routine, and writes the solution back to a file. When reading the mesh, we check it for duplicated vertices, topological errors, and inverted elements, and we compute the vertices on the boundary of the mesh.

The average inverse mean-ratio objective function requires that a value of plus infinity be returned whenever the volume constraints are not satisfied. Therefore, if the volume of at least one element is smaller than 1.0×10^{-14}, we consider the volume constraints to be violated, and the objective function is set to plus infinity. This strategy is reasonable when the mesh is well scaled because the objective function becomes very large.

The gradient and Hessian of the objective function are calculated analytically by assembling the gradients and Hessians for each element function into a vector and symmetric sparse matrix. In order to facilitate the assembly of the Hessian matrix, once the sparsity pattern is obtained, an additional vector is calculated that tells the offset into the Hessian matrix data vector where the element Hessians are to be accumulated. The gradient and Hessian of the elements with respect to vertices fixed on the boundary of the mesh are ignored.

The code for calculating the gradient of the element function uses the reverse mode of automatic differentiation [6, 16]. The code was written and refined by hand to eliminate unnecessary operations, resulting in a more efficient gradient evaluation. The Hessian calculation uses the forward mode of differentiation on the gradient evaluation. The resulting code was written by using matrix-matrix products for efficiency. AMPL was used to verify the correctness of the analytic gradient and Hessian evaluations.

Table 2 presents the results obtained when using the KNITRO libraries for the optimization solver. Two versions of the code were run: one where the Hessian matrix was directly provided to the code in the (i, j, k) format, the "Hessian" column, and the other where a routine for computing Hessian-vector products was supplied, the "Product" column. The routine stores the upper triangular part of the Hessian matrix in a block compressed sparse row format, where each block corresponds to a vertex-vertex interaction in the original mesh, and performs a Hessian-vector product using this structure. The version of KNITRO using Hessian-vector products is faster than when the Hessian matrix is supplied and significantly faster on the larger examples. Moreover, the block upper triangular structure stores only one index per block, instead of two indices per variable, resulting in some memory savings.

In order to reduce the computational time further, the vertices and elements in the initial mesh were re-

Table 2: Results using KNITRO libraries with original vertex order.

Mesh	Variables	AMPL	Hessian	Product
gear	780	1.87	0.07	0.07
foam5	867	2.34	0.11	0.11
hook	1,200	3.15	0.12	0.11
duct20	1,146	3.12	0.10	0.10
duct15	2,895	7.96	0.26	0.25
duct12	6,906	21.59	0.64	0.63
duct10	13,440	49.27	1.39	1.33
duct8	26,214	124.81	3.65	3.07
duct4	425,952	-	205.11	148.46
duct2	3,323,229	-	-	-

Table 3: Results using KNITRO libraries with breadth-first search vertex order.

	Original		Reordered	
Mesh	Hessian	Product	Hessian	Product
gear	0.07	0.07	0.07	0.07
foam5	0.11	0.11	0.11	0.11
hook	0.12	0.11	0.12	0.11
duct20	0.10	0.10	0.10	0.10
duct15	0.26	0.25	0.25	0.25
duct12	0.64	0.63	0.59	0.58
duct10	1.39	1.33	1.18	1.16
duct8	3.65	3.07	2.95	2.49
duct4	205.11	145.98	134.06	81.74
duct2	-	-	-	-

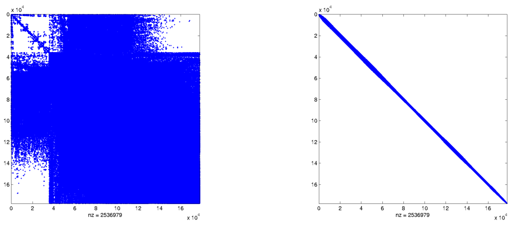

Figure 2: Sparsity pattern of the Hessian matrix for the duct8 mesh with original ordering (left) and the breadth-first search ordering (right).

ordered by using a breadth-first search ordering [25] to improve the locality of reference prior to applying KNITRO. Figure 2 shows the sparsity pattern for the Hessian matrix of the original and reordered mesh for the duct8 problem. In particular, the ordering starts by selecting the (boundary) vertex farthest from the origin as a starting point. A breadth-first search of the vertices in the mesh is then performed. The order in which the vertices were visited is reversed, as in the reverse Cuthill-McKee ordering [9], to obtain a symmetric permutation of the vertices for the optimization problem. The elements in the mesh are then reordered according to when they are referenced by the vertices. Other orderings can be applied that may give rise to further improvements in performance [17].

Table 3 presents the results on the original and reordered meshes. The time for the reordered test problems includes the cost of computing the breadth-first search and reordering the problem data. The savings attributed to reducing the bandwidth of the matrices is considerable, especially on the larger optimization problems.

To further improve performance, we would like to apply an appropriate preconditioner in the conjugate gradient method used within the optimization solver. For the mesh optimization problems we know that the diagonal blocks of the Hessian matrix are positive definite [22, 23], so a block Jacobi preconditioner can be applied. Unfortunately, the available version of the KNITRO libraries does not currently allow the user to provide a preconditioner. Therefore, a simple inexact Newton method [18, 24] with an Armijo linesearch [1] was built on the same framework to solve the mesh optimization problems.

In particular, given x^k, the algorithm computes a direction d^k by solving the symmetric system of linear equations

$$\nabla^2\theta(x^k)d^k = -\nabla\theta(x^k)$$

by applying a conjugate gradient method with a block Jacobi preconditioner [25]. Since the Hessian can be indefinite, the conjugate gradient method may terminate with a direction of negative curvature. In such a case, the base of the direction is used as the starting point for the linesearch. The Armijo linesearch finds the smallest nonnegative integer m such that

$$\theta(x^k + \beta^m d^k) \le \theta(x^k) + \sigma\beta^m\nabla\theta(x^k)^T d^k,$$

where $0 < \sigma < \frac{1}{2}$ and $0 < \beta < 1$ are constants. The iterate is then updated with the rule $x^{k+1} = x^k + \beta^m d^k$, and a new direction is computed. The algorithm terminates when $\|\nabla\theta(x^k)\|_2$ is less than 1.0×10^{-6}.

Table 4: Results using block Jacobi preconditioner.

Mesh	Variables	AMPL	KNITRO	Newton
gear	780	1.87	0.07	0.06
foam5	867	2.34	0.11	0.09
hook	1,200	3.15	0.11	0.09
duct20	1,146	3.12	0.10	0.07
duct15	2,895	7.96	0.25	0.19
duct12	6,906	21.59	0.58	0.45
duct10	13,440	49.27	1.16	0.88
duct8	26,214	124.81	2.49	1.78
duct4	425,952	-	81.74	46.41
duct2	3,323,229	-	-	324.92

The conjugate gradient method terminates if the system of equations is solved to within a specified tolerance, if a direction of negative curvature is encountered, or if 100 conjugate gradient iterations have been performed. In particular, the conjugate gradient implementation terminates when

$$\|\nabla^2\theta(x^k)d^k + \nabla\theta(x^k)\|_2 \leq 10^{-2} \times \|\nabla\theta(x^k)\|_2.$$

That is, the relative tolerance is used for the termination test.

Table 4 presents the final results obtained by using the reordered meshes with AMPL, the Hessian-vector product version of KNITRO, and the inexact Newton method with a block Jacobi preconditioner. As expected, the preconditioner helps to further reduce the computational time. The results are less dramatic on the smaller problems because of the fixed time required to set up the problem; the improvement from preconditioning is more dramatic on the larger examples. In particular, the preconditioned code was able to compute a solution to the duct2 problem; the KNITRO libraries were terminated after an hour of computational time without finding a solution.

4. Conclusions

This article discussed an optimization problem for finding the optimal vertex positions in a mesh according to the average inverse mean-ratio metric. Also presented was a computational study of three techniques to improve solver performance on the unconstrained version of the problem. Modifying the data structures, reordering the problem data, and preconditioning the iterative method can significantly reduce the computational time, especially for large instances.

However, developing a framework for a specific problem and validating the result is a time-consuming task recommended only if performance is crucial. Tweaking the implementation to make the code more efficient is another critical step requiring a significant time investment. After going through this process for the mesh optimization application, we observed that solving the problem through AMPL was 25–70 times slower than using the preconditioned inexact Newton code and could consume over 120 times the memory.

We prefer using solver libraries rather than implementing our own numerical optimization algorithms. However, the design of the library is important to achieve high performance. Matrix-free methods are desirable because they allow the application developer to determine the data structures used to store the matrices and to identify ways to efficiently perform the required matrix-vector products, which can reduce both computational time and memory requirements. Moreover, allowing the user to specify a preconditioner for the iterative methods employed can be beneficial; however, an appropriate strategy to precondition constrained optimization problems is an open question. Reordering the matrices to reduce the bandwidth can also result in significant improvements in time for unconstrained optimization problems. The correct reordering to use for constrained problems is likely algorithm dependent and also an open question.

We remark that the fluid dynamics application in the introduction solved a constrained version of the mesh optimization problem, where the constraints restrict the vertices on the boundary to planes or spheres, rather than fixing them in position. We were able to construct a model for this application because we knew the geometry of the problem. A general-purpose tool for constrained mesh optimization problems would require either knowledge of the geometry or adding a strategy to the framework to uncover simple geometric objects such as planes and ellipses. The latter strategy has not yet been implemented in our framework but is a planned extension. Once constraints are added to the framework, we can investigate the effect of the reordering and matrix-free strategies on the solution time when using the KNITRO libraries. Preconditioning the constrained case and making effective use of a good starting point in an interior-point method are open issues. However, this work is essential for using mesh optimization with the original fluid dynamics application where the particles move as a function of time.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Lori Freitag-Diachin, Patrick Knupp, and Suzanne Shontz introduced me to the mesh shape-quality optimization problem and the inverse mean-ratio metric; David Gay provided the elegant version of the volume calculation used in the AMPL models; Paul Hovland wrote the initial version of the analytic gradient of the element function; Richard Waltz kindly provided the version of the KNITRO libraries used; and Lin Zhang, S. Balachandar, and Paul Fischer provided the meshes and performance data for the computational fluid dynamics example.

REFERENCES

[1] L. Armijo, *Minimization of functions having Lipschitz-continuous first partial derivatives*, Pacific J. Math., 16 (1996), pp. 1–3.

[2] I. Babuška and M. Suri, *The p and h-p versions of the finite element method, basic principles and properties*, SIAM Rev., 36 (1994), pp. 578–632.

[3] R. E. Bank and R. K Smith, *Mesh smoothing using a posteriori estimates*, SIAM J. Numer. Anal., 34 (1997), pp. 979–997.

[4] M. Berzins, *Solution-based mesh quality for triangular and tetrahedral meshes*, in *Proceedings of the Sixth International Meshing Roundtable*, Sandia National Laboratories, 1997, pp. 427–436.

[5] M. Berzins, *Mesh quality – geometry, error estimates or both?*, in *Proceedings of the Seventh International Meshing Roundtable*, Sandia National Laboratories, 1998, pp. 229–237.

[6] C. H. Bischof, P. D. Hovland, and B. Norris, *Implementation of automatic differentiation tools*, in *Higher-Order and Symbolic Computation*, 2004, to appear.

[7] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 2002.

[8] R. Byrd, M. E. Hribar, and J. Nocedal, *An interior point method for large scale nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 877–900.

[9] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in *Proceedings of the 24th National Conference ACM*, ACM Press, 1969, pp. 157–172.

[10] M. O. Deville, P. F. Fischer, and E. H. Mund, *High-Order Methods for Incompressible Fluid Flows*, Cambridge University Press, Cambridge, 2002.

[11] R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Brooks/Cole–Thomson Learning, Pacific Grove, California, second edition, 2003.

[12] L. Freitag and P. Knupp, *Tetrahedral mesh improvement via optimization of the element condition number*, Internat. J. Numer. Methods Engrg., 53 (2002), pp. 1377–1391.

[13] L. Freitag, P. Knupp, T. Munson, and S. Shontz, *A comparison of optimization software for mesh shape-quality improvement problems*, in *Proceedings of the Eleventh International Meshing Roundtable*, pp. 29–40, 2002, Sandia National Laboratories.

[14] L. Freitag and C. Ollivier-Gooch, *A comparison of tetrahedral mesh improvement techniques* in *Proceedings of the Fifth International Meshing Roundtable*, pp. 87–100, 1996, Sandia National Laboratories.

[15] L. Freitag and C. Ollivier-Gooch, *A cost/benefit analysis for simplicial mesh improvement techniques as measured by solution efficiency*, Internat. J. Comput. Geom. Appl., 10 (2000), pp. 361–382.

[16] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, 2000.

[17] H. Han and C. Tseng, *A comparison of locality transformations for irregular codes*, in *Proceedings of the Fifth International Workshop on Languages, Compilers, and Run-time Systems for Scalable Computers*, Springer-Verlag, New York, 2000, pp. 70–84.

[18] C. T. Kelley, *Solving Nonlinear Equations with Newton's Method*, SIAM, Philadelphia, 2003.

[19] P. Knupp, *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part I – A framework for surface mesh optimization*, Internat. J. Numer. Methods Engrg., 48 (2000), pp. 401–420.

[20] P. Knupp, *Achieving finite element mesh quality via optimization of the Jacobian matrix norm and associated quantities, Part II – A framework for volume mesh optimization and the condition number of the Jacobian matrix*, Internat. J. Numer. Methods Engrg., 48 (2000), pp. 1165–1185.

[21] A. Liu and B. Joe, *Relationship between tetrahedron quality measures*, BIT, 34 (1994), pp. 268–287

[22] T. S. Munson, *Mesh shape-quality optimization using the inverse mean-ratio metric*, Preprint ANL/MCS-P1136-0304, Argonne National Laboratory, Argonne, 2004.

[23] T. S. Munson, *Mesh shape-quality optimization using the inverse mean-ratio metric: Tetrahedral proofs*, Technical Memorandum ANL/MCS-TM-275, Argonne National Laboratory, Argonne, 2004.

[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.

[25] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, second edition, 2003.

[26] *CUBIT 8.1 Mesh Generation Toolkit*, Sandia National Laboratories, Albuquerque, 2003.

[27] M. Shephard and M. Georges, *Automatic three-dimensional mesh generation by the finite octree technique*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 709–749.

[28] J. Shewchuk. *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, in *Proceedings of the First Workshop on Applied Computational Geometry*, ACM, Philadelphia, May 1996, pp. 124–133.

[29] J. Shewchuk, *What is a good linear element? Interpolation, conditioning, and quality measures*, in *Proceedings of the Eleventh International Meshing Roundtable*, Sandia National Laboratories, 2002, pp. 115–126.

[30] L. N. Trefethan, *Spectral Element Methods in MATLAB*, SIAM, Philadelphia, 2000.

[31] R. J. Vanderbei, *LOQO user's manual – Version 4.05*, Technical report, Princeton University, Princeton, 2000.

[32] R. J. Vanderbei and D. F. Shanno, *An interior-point algorithm for nonconvex nonlinear programming*, Comput. Optim. Appl., 13 (1999), pp. 231–252.

[33] R. Waltz and J. Nocedal, *KNITRO user's manual – Version 3.1*, Technical Report 5, Northwestern University, Evanston, 2003.

[34] L. Zhang, S. Balachandar, P. Fischer, and T. Munson, *Private communication*, December 2004.

Bulletin

1. Event Announcements

19th International Symposium on Mathematical Programming
July 30 – August 4, 2006
Federal University of Rio de Janeiro, Brazil
`http://www.lncc.br/~ismp2006`

The International Symposium on Mathematical Programming is a scientific meeting held every 3 years on behalf of the Mathematical Programming Society.

The 19th Symposium will be hosted by the Systems Engineering and Computer Science Program of the Graduate School and Research in Engineering, COPPE/UFRJ, and will be held on the campus of the Federal University of Rio de Janeiro in Praia Vermelha, Rio de Janeiro, RJ, Brazil, from July 30 to August 4, 2006.

2. Other Announcements

SeDuMi

The Advanced Optimization Laboratory at McMaster University, Canada (http://www.cas.mcmaster.ca/˜oplab), is glad to announce that it takes over the maintenance and development of the SeDuMi package originally created by Jos F. Sturm, who passed away in October, 2003.

The new SeDuMi website is at http://sedumi.mcmaster.ca. It currently contains a Downloads section with the latest SeDuMi releases, documentation, test sets and related papers, FAQ and a user forum. Users can provide feedback and share their opinion and success stories about the software, request new features, report bugs, etc. at the forum.

Currently, SeDuMi works in Matlab environment on Windows and Unix/Linux systems. Experimental Mac builds and instructions on how to compile SeDuMi under Mac OS X are under development. Updates for Matlab 7 are going to be released in the near future. The maintainers also plan to provide Windows builds optimized for Pentium IV and Athlon processors.

New features are planned to be added based on user requests and recent publications.

Tams Terlaky (**terlaky@mcmaster.ca**),
Imre Plik (**poliki@mcmaster.ca**),
Oleksandr Romanko (**Romanko@mcmaster.ca**).

Chairman's Column

This is the first issue of *Views and News* to appear since the Eighth SIAM Conference on Optimization that took place in Stockholm, Sweden from May 15-18, 2005. On behalf of all the participants I would like to thank Henry Wolkowicz, Anders Forsgren and the rest of the organizing committee for the wonderful job they did in putting on the meeting. Those of you who were there don't need me to report on how successful it was, but for others who could not go I will give a few highlights. The meeting was extremely well attended and came within a few registrants of tying as the largest SIAM Optimization Conference to date. The meeting was held in the charming "Norra Latin" building of the Stockholm City Conference Center. In addition to being well located and ideally suited to the size of the meeting, this renovated school offered ample gathering space so that participants could easily meet when not attending sessions. The conference dinner at the beautiful Stockholm City Hall offered everyone the chance to pretend for a moment that they were attending a Nobel Prize banquet. As co-chair (with Sven Leyffer) of the organizing committee for the next Optimization Conference, to be held in 2008, I feel that we have a very tough act to follow. We are currently working with SIAM to finalize the location of the meeting, which will be held in the United States. Please feel free to contact either Sven or me if you have any comments about the meeting held in Stockholm and/or suggestions for the 2008 meeting.

On a much sadder note this is also the first issue of *Views and News* since the deaths of George Dantzig and Leonid Khachiyan shortly before the SIAM Optimization Conference was held. I would like to thank Walter Murray and Mike Todd for writing the appreciations of George and Leo that appear in this issue. Much has already been and will continue to be written about the contributions of these two scientists, but I would like to add a few words of my own. It is impossible to overstate the influence that Dantzig and Khachiyan had on the field of mathematical optimization. As Walter writes, George was the "Father of Linear Programming", and by logical extension the father of mathematical programming as we know it. Of course Dantzig is best known for linear programming and the simplex method, but it is important to remember that from very early on his interests included more general optimization problems. The 1963 edition of *Linear Programming and Extensions* includes complete treatments of constrained convex programming (via cutting planes and "Wolfe's Generalized Program") and linear discrete optimization (via Gomory cuts), among many other topics. These algorithmic approaches may no longer be considered the best available (although there has been a resurgence of interest in Gomory cuts in recent years), but they certainly demonstrate the breadth of Dantzig's interests and his understanding of the importance of nonlinearities and discrete variables in practical applications.

Dantzig's work was always motivated by the desire to solve problems, and in particular the desire to solve large problems. He was concerned with the theoretical correctness of algorithms, as characterized by finiteness or convergence, but was primarily interested in what actually worked in practice on large-scale instances. Khachiyan's interest was also in solving large-scale problems, but he brought to the field of continuous optimization the viewpoint of a theoretical computer scientist. A "provably" good algorithm for large-scale problems was one whose complexity to solve, or approximately solve, problems did not grow too fast as problems got larger; in particular one whose complexity was *polynomial-time* in the problem size. Khachiyan's analysis of the ellipsoid algorithm for linear programming was a revolutionary breakthrough. As described in Mike Todd's appreciation, Khachiyan was forced to think about linear programming in a completely new way in order to frame his results using the vocabulary of computational complexity in the rational model. Although the ellipsoid algorithm was not successful in practice it became a very important tool in complexity analysis for combinatorial problems, as described in the classic text *Geometrical Algorithms and Combinatorial Optimization* by Grötschel, Lovász and Schrijver. The conflict between the theoretical and practical views (the simplex method being excellent in practice but exponential in the worst case, while the ellipsoid algorithm was polynomial in the worst case but impractical on real-world problems) also led to a great deal of research, culminating with the discovery of interior-point algorithms that were both theoretically and practically efficient. The viewpoint of Narendra Karmarkar, like Khachiyan's, was clearly that of a theoretical computer scientist. Without the motivation of complexity analysis it seems very unlikely that anyone would have thought of applying out-of-favor techniques from nonlinear optimization to try to solve large-scale linear programming problems.

In addition to their enormous scientific accomplishments both Dantzig and Khachiyan were well known for their warmth and generosity, as described by Walter and Mike in their articles. Having been George's student and shared research interests with Leo I can confirm these widely-held views with my own experiences. Standards of professional conduct have a tendency to propagate from the "top down" in scientific communities, and we have been very fortunate to have researchers like Dantzig and Khachiyan as models for both mathematical innovation and personal interaction. Their influence will continue to be felt as long as optimization exists as a research discipline.

Kurt M. Anstreicher, SIAG/OPT Chair
Department of Management Sciences
University of Iowa
S210 PBB Iowa City, IA 52242,
USA
kurt-anstreicher@uiowa.edu
`http://www.biz.uiowa.edu/faculty/anstreicher`

Comments from the Editor

The current issue of the SIAM/Optimization Views-and-News (Volume 16, Numbers 1-2) features the topic *Algebraic Methods for Integer Programming*, which was especially edited for us by Karen Aardal. The three expository articles describe, in an accessible way, some of the most recent and exciting developments in the area of Integer Programming. I would like to thank Karen Aardal, our guest editor, and the contributing authors (Karen Aardal, Daniel Bienstock, Kevin Woods, and Ruriko Yoshida) for their efforts. As editor, I am also grateful to João Soares who kindly reviewed the three articles.

This issue, however, is not just on integer programming. We also have a very interesting article on mesh optimization by Todd Munson.

And, last but not the least, I am very pleased to publish in this issue of our newsletter tributes to two great optimizers recently passed away: George Dantzig and Leonid Khachiyan. I would like to express my deep gratitude to Walter Murray and Michael Todd for having written such great and inspired personal appreciations.

Contributions to the newsletter such as announcements of forthcoming events or releases of new books and software packages are always extremely welcome.

Luís N. Vicente, Editor
Department of Mathematics
University of Coimbra
3001-454 Coimbra
Portugal
lnv@mat.uc.pt
`http://www.mat.uc.pt/~lnv`