European Journal of Operational Research 253 (2016) 543–556

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

Origin and early evolution of corner polyhedra

Ralph Gomory*

New York University, Stern School of Business, United States

ARTICLE INFO

Article history:
Received 27 March 2015
Accepted 1 March 2016
Available online 8 March 2016

Keywords:
Integer programming
Cutting
Linear programming
Corner polyhedra

ABSTRACT

Corner Polyhedra are a natural intermediate step between linear programming and integer programming. This paper first describes how the concept of Corner Polyhedra arose unexpectedly from a practical operations research problem, and then describes how it evolved to shed light on fundamental aspects of integer programming and to provide a great variety of cutting planes for integer programming.

1. Introduction and background

1.1. Intent of this article

This article is a personal account of my experiences with Corner Polyhedra and some closely related integer programming problems. It will not be a survey of the related literature, a survey which, because of my intermittent connection with the subject, I really could not write in an informed and balanced way.[1]

The article starts by describing work on the practical problems of paper mills. The necessity of dealing with the very large size of these problems motivated the invention of what is now called column generation. Then the results obtained using these methods turned out to have a completely unexpected periodicity.

Explaining that periodicity led through various stages of understanding to the creation of the polyhedra that I named Corner Polyhedra. We will see how the Corner Polyhedra then took on a life of their own, giving insight into the structure of integer polyhedra and generating new families of cutting planes for general integer programming.

Throughout I will do my best to describe the surroundings and motivation that drove this evolution and to make the various steps as clear as possible and illustrate them by examples. I will also occasionally point to directions which seem to me to have unrealized possibilities.

1.2. Background: applied mathematics and operations research

In the 1950's Operations Research was a new and exciting part of Applied Mathematics. It was appealing to me because it promised to extend the reach of Mathematics beyond the traditional fields of Science and Engineering and closer to ordinary life. And that did happen.

But in addition, many times, Operations Research work motivated by practical needs, has also turned out to be mathematically beautiful.

Many believe that Applied Mathematics, and especially Operations Research, is mainly a routine use of mathematics. Many believe that operations researchers find a problem, apply to it some well understood piece of mathematics, and the answer comes out. That ends it, the problem is solved.

This certainly can happen, but often applied work is much more complicated than that. In applied work finding a way to formulate a problem mathematically can be difficult in itself. Then if you do succeed in finding a mathematical formulation, its sheer size and complexity may overwhelm standard approaches. You may have to split out tractable parts and leave the rest, or you may have to find a way to approximate, or you may have to invent.

Sometimes you may succeed in all this only to find that your hard won solution is met with hostility by those who might be affected by it, or alternatively, you may be fortunate and find that those affected by your work are surprisingly eager to adopt it for reasons quite unconnected with what you have done.

And, every now and then, you may turn up something unexpected, something you stumble across while pursuing something

* Tel.: +1 9142388522.
E-mail address: gomory@sloan.org

[1] This paper is based on my lecture "Forty Years of Corner Polyhedra" given on July 11, 2012 at EURO XXV. I want to thank the many researchers who have sent me their papers over the years, including the many years that I have spent away from integer programming, being engaged in other work. Trying to mention their work and place it in the proper context would be a very large and very worthwhile enterprise, but I do not attempt it in this paper.

http://dx.doi.org/10.1016/j.ejor.2016.03.001
0377-2217/

else. When I think of this sort of thing I always think of Christopher Columbus.

Although Columbus lived in a world very different from ours, he had some problems that resemble those we have today. In modern terms we would say that Columbus had major difficulties getting his project funded, and then, after his plan to reach China and India by sailing west was finally accepted, he failed to get there.

Columbus promised to find a new route to the Indies. But Columbus didn't find a new route to the Indies. Instead he discovered a whole new world. This too can happen.

1.3. Some modern explorers

About 370 years after the Queen of Spain sent Columbus off toward the Indies, a modern ruler, IBM, sent off another smaller group of explorers. Our group of explorers was chartered to see if the mathematical methods of Operations Research could find and conquer rich new territories for computers.

Among our explorers were Benoit Mandelbrot, Paul Gilmore, and T.C. Hu. Later we were joined by Philip Wolfe and Ellis Johnson.

Columbus' expedition embarked in three ships, the Nina, the Pinta, and the Santa Maria. Our smaller expedition also relied on three ships. Ours were named Linear, Dynamic, and Integer Programming. As it turned out, we needed all three for our voyage.

2. Origins of column generation

2.1. The stock cutting problem

Our little group within IBM's Research Division, was aware of the general stock cutting problem. This is the problem of starting with a stock of large pieces of some material and then cutting those large pieces into needed quantities of smaller sizes while creating as little waste as possible. We understood that a great variety of stock cutting problems could be formulated as linear or integer programming problems; so we wondered if there was something in this area that we could put into actual use.

As a first step we tried a problem we had heard about that involved cutting up big steel girders for bridges; but although we could formulate the problem mathematically, the data from real bridges gave us integer programming problems that were way beyond the scale that anyone at that time could handle.

2.2. The paper trim problem – linear programming

Next Paul Gilmore and I took a look at a different stock cutting problem, the paper trim problem, the problem of cutting the very wide rolls of paper that paper mills produce into the smaller width rolls that people actually use. We had heard that there were special aspects of the paper trim problem that might allow us to use ordinary linear programming instead of integer programming; if true, that would make the problem more tractable.

In the paper trim problem, as in the bridge problem, the actual numbers matter. So here is a description, based on our later experience, of a typical paper trim problem.[2]

Paper comes streaming out of an enormous paper machine in a paper mill and is rolled up on metal spindles. The paper the machine makes has a fixed large width W, which depends on the paper machine. A typical width W is 200 inches or more. The mill's customers want rolls of paper in a variety of much smaller sizes, thirty different customer widths w_j of 20 to 80 inches would be typical. So the mills are obliged to cut up the wide rolls they manufacture into the quantities b_j of these smaller rolls that the customers want.

To produce the right quantities of the smaller rolls, the mills have to cut up their wide rolls in many different ways. Each way to cut up one wide roll is called a cutting pattern. The *i*th cutting pattern is a list A_i that gives the number $a_{i,j}$ of rolls of a width w_j that the pattern produces.

It was well known that the paper trim problem could be formulated as an integer programming problem. In this standard formulation x_i is the (integer) number of times the *i*th cutting pattern is used, n is the number of different cutting patterns A_i that are available (this is often a very large number), m is the number of different customer widths w_j that are demanded, and the goal is to choose integers x_i $(i = 1,\ldots,n)$ that fill the m customer orders b_j while minimizing the cost.

Cost is taken to be the number of wide rolls required to fill all the orders. So here is a standard formulation:

$$\text{Minimize } V = \sum_{i=1}^{i=n} x_i \quad \text{subject to} \tag{1}$$

$$\sum_{i=1}^{i=n} a_{i,j} x_i \geq b_j \quad \text{for} \quad j = 1, \ldots, m \tag{1A}$$

If we choose, we can add non-negative slack variables to (1A) to produce a formulation free of inequalities.

Now (1) and (1A) together are a very straightforward integer linear programming problem. The only question is: is it too large to handle?

What we had heard about the paper industry was that it was acceptable for the customer requirements, the b_j not to be exact requirements. It was acceptable to produce more than the amounts the customer asked for by a few percent, the customers were willing to take the extra rolls.

What that meant to us was this: if the *linear* programming solution to (1) and (1A) gave us some non-integer x_j, cutting patterns that were used a non-integer number of times, perhaps we could just round up those x_j and the customer would accept the extra rolls that were generated.

That sounded promising.

2.3. Size still a problem

However, even though it was now possible that we were dealing with an ordinary linear programming problem, our mathematical formulation still presented a difficulty: the enormous number of possible cutting patterns. With realistic paper industry problems, with for example 30 different customer widths to choose from, the number of cutting patterns, and therefore the number of columns in the linear programming problem, could easily run into many millions.

Now solving a linear program with millions of columns was well beyond what even the largest computers of that time could do, while the paper mills at that time were yet to purchase even a small computer.

So we had to think hard about how to get the problem down to something that could run on a small computer, and then hope that the cost savings our calculation might produce would cover its cost.

We thought hard about what the simplex method actually does in (1) and (1A) and eventually found an approach.

2.4. A starting solution

Suppose we start the simplex calculation of (1) and (1A) using an arbitrarily chosen set of cutting patterns A_i that we take as an

[2] Gilmore and Gomory (1963) has a detailed description of an actual problem as an appendix.

initial basis. To form a first basic feasible solution we need only m cutting patterns if there are m different customer widths. In addition to this initial set we added to our starting problem a few more arbitrarily chosen cutting patterns which might turn out to be useful for improving from our initial basis.

Note that In our linear programming formulation (1) and (1A) the column for each cutting pattern A_i has in addition to the m elements $a_{i,j}$ that describe the rolls it produces, an entry -1 in the objective function row (1) that reflects the wide roll it is using up.

Let M be the matrix that contains a column for each cutting pattern in the basis, all of them with -1's in the row that represents the objective function. M also has an additional first column for the V that represents the objective function. This is a unit column, with 1 at the top row and then zeroes. M is a (m+1) x (m+1) matrix.

The simplex method starts by finding the inverse M^{-1} of M. Then, using $M^{-1}$and following the simplex method, we transform the entire starting matrix into a first basic feasible solution. The extra cutting patterns that we are carrying along are now columns associated with non-basic variables.

2.5. Improving the starting solution

If we look at the top row of the entire transformed matrix, this is the new cost row. The top entry in every transformed cutting pattern column is the result of taking the dot product of the top row of M^{-1} with the cutting pattern column (-1, A_i).). The top row of M^{-1} has a 1 in the column representing V, and then a string P of m non-negative numbers (p_1, $p_{2,\ldots,}p_m$); so the dot product with the column containing A_i is $(1, P) \cdot (-1, A_i) = (-1, P \cdot A_i) = (-1, p_1a_{i1}, p_2a_{i,2},\ldots,p_ma_{i,m})$

The p_i are the shadow prices that reward the cutting pattern for the customer widths $a_{i,j}$ it produces, while the -1 in the V column penalizes it for using up the wide rolls that the mill produces.

The simplex method tells us that if the cost row entry of a non-basic cutting pattern column is positive, that is if the value of what it produces outweighs its cost, then using that column to move to a new simplex basis will give an improved solution.

So the problem of finding an improved solution has become this: can we find a new cutting pattern A_i such that its contribution outweighs its cost? Since the non-basic columns were arbitrarily chosen, and could be any of the millions of possible cutting patterns, we are asking this: is there, among all possible cutting patterns A_i, an A_i that will make a positive contribution given the current values (p_1, $p_{2,\ldots,}p_m$) of the shadow prices. Is there an A_i with $P \cdot A_i > 1$?

2.6. The knapsack problem

While any improving cutting pattern would be welcome, we would have an especially warm welcome for the cutting pattern that makes the biggest positive contribution. So let us shift gears slightly and look for that. So we are now looking for the cutting pattern A_i that maximizes $P \cdot A_i$.

Although the words we are using are different, we now have a classical knapsack problem. In the knapsack problem we have an assortment of objects with different values and different weights. The knapsack problem is to find the most valuable assortment of objects to put into the knapsack without exceeding a specified total weight.

Our problem here is to find the most valuable cutting pattern given the current prices p_j for each customer width w_j produced. Our constraint here is not a specified total weight, it is a total width. To qualify as a cutting pattern, the collection of widths in a cutting pattern must add up to a total width not exceeding W, the width of the roll the paper machine produces.

So, in looking for the cutting pattern A_i that maximizes $P \cdot A_i$, we have, in different words, a knapsack problem.

This is good news because there are many simple ways, including standard Dynamic Programming, to solve knapsack problems.

2.7. An algorithm

All this enabled an algorithm that no longer required millions of columns.

After obtaining the initial basis as described in Sections 2.4 and 2.5, we take as the value for each customer width w_j the shadow price p_j, and solve our constrained width knapsack problem using dynamic programming. Then we take the winning cutting pattern as a new column for entry into the next basic feasible solution of the simplex method.

We can repeat this process over and over, generating new columns, going to new and improved feasible bases, until we have a basis with the property that the best new cutting pattern does not make a positive contribution. That means that there are no cutting patterns that will improve our current basis. We have the best possible basis, the best possible collection of cutting patterns.

We had found a way to deal with the millions of cutting patterns. We now had all the elements for an algorithm.

Carol Shanesy, our wonderful programmer, soon gave us a FORTRAN program that combined this method of generating new cutting patterns with the steps of the simplex method. We had an algorithm that actually ran.

3. Practical progress

3.1. From algorithms to paper mills

At this point we felt we had done what we could do from a distance; we had an approach to the integer requirement by rounding, and an approach to the problem of the millions of patterns through column generation. It was time to see some real paper mills and see what could actually be accomplished. Did our mathematical model fit what was really going on? Would the rounding yield acceptable results? Or were there other conditions that were real and important that we did not include in our formulation.

Even if our formulation did match the actual situation, or could be changed to match it, would our methods save enough paper to make the use of our methods worthwhile? After all, the mill would have to buy a computer.

Clearly the next step was to find a way into the land of real paper mills. Fortunately we had guides.

Our guides were always IBM salesmen, people accustomed to selling IBM equipment to paper mills. Since at that time paper mills did not have computers, our guides usually sold the mills things like time clocks or accounting machines. The salesmen liked the idea of having something new to sell, so they were willing to work with us.

So Paul Gilmore and I set out. We hoped, as a minimum, to learn more about the problem by visiting actual paper mills. We also hoped, if our approach still seemed reasonable when confronted by the realities of actual mills, to find a mill that would actually try out our ideas.

Paper mills were then, and probably still are now, very impressive places. At a paper mill big hunks of wood are thrown in at one end of a paper making machine, which can easily be 400 feet long, and a wide stream of paper comes out at fifty miles an hour at the other end. As it emerges, this stream of paper is rolled up on a succession of big metal spindles and is ready for the cutting process.

We soon learned that at the mills there were some very special people who cut up the wide rolls. They decided how to cut by an intuitive feel developed by experience, lots of experience. There were some people who did this job well, and there were others who did not; it was a skill not everyone could develop. Some mill managers were worried; many workers who did a good job of cutting were getting older, it was not clear they could easily be replaced when they retired.

Partly because of this concern there were a few mills that were willing to try what was for them a fairly painless experiment. They gave us their customer orders for a month or so and asked us to come back with our calculated results. They could then see how our results compared with what they had actually done.

3.2. Computing and early results

We took their data back to our home base at IBM Research and started running calculations. After a few false starts our programs ran on their data, and what came out looked good to us. When we had enough results we sent them back to our IBM salesmen who then took them over to the mills.

At the mills there was agreement that rounding was not a problem and that our solutions did reduce waste. Usually our solutions saved only a few percent, but sometimes more. But even a few percent of the output of a paper mill is a lot of paper. Though savings varied quite a bit from one mill to another there was often enough saved to make it economically attractive for the mill, provided the computation could be done at the mill and on a small computer.

So this was starting to be serious.

3.3. Reducing running time and other progress

The hard problem was running time.

We knew that to succeed we needed to cut down running times. The mills needed to get the results that we were getting on our big computer at IBM Research on a small computer that was affordable to the mills, and the results had to come out reasonably quickly.

We worked to reduce running time and looked at lots of examples. We learned by trial and error. We learned to avoid cutting patterns that included both rolls for which there was high demand and rolls for which there was low demand. We learned when it was time to cut off the calculation and to stop, even though the result was still improving slowly etc.

In our calculations we found that we were spending much more time solving the knapsack problems that found a new cutting pattern, than on the next linear programming step that used that new cutting pattern, so Paul and I worked hard at ways to improve the knapsack calculation itself.

In the end what we did worked. We got the running time down to the point where paper mills started to buy computers for the first time. Paul Gilmore and I were even named “IBM Science Salesmen of the Month”, for May 1961 or possibly May 1962. Being IBM science salesman of the month is not quite the same as winning the Nobel Prize, but we felt good about it nevertheless.

Eventually our methods and improvements on them became very widely used in paper mills.

We also extended our methods to classes of two dimensional problems (glass is an example) and wrote an article describing fast methods for the knapsack problem papers (Gilmore & Gomory, 1961, 1963; 1965). The idea of column generation became quite widespread. And for our paper that described many of these things Paul Gilmore and I were awarded the Lanchester prize of the Operations Research Society in 1963.

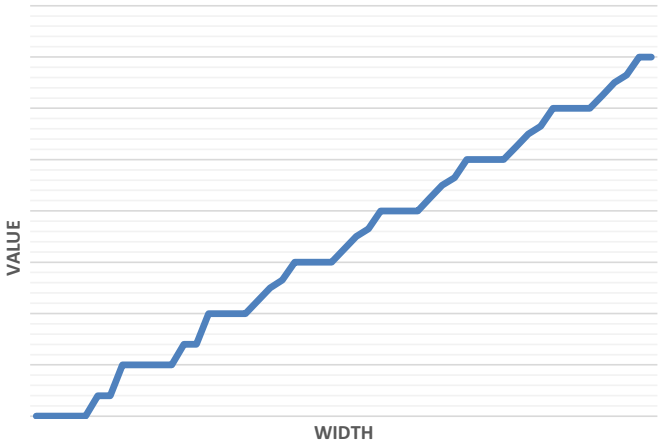

Fig. 1. Periodicity.

4. Encountering the unexpected: periodicity appears

4.1. Data and periodicity

However while we were visiting paper mills and glass plants and improving and extending our methods, we were also collecting lots of data. One of the things we collected data on was the effect on waste of the width of the paper machine[3]. We also looked at the knapsack problems separately from the rest of our algorithm to see how much individual cutting patterns improved with increasing W.

Here we show in somewhat idealized form a typical knapsack graph (Fig. 1).

The values (shadow prices) and widths of the individual rolls are fixed. The graph shows the total value of the small rolls that can be cut from one wide roll as a function of machine width W. We can see that the total value of the knapsack increases steadily with increasing width, as it should, but in addition, we see that, after an initial period of irregular increases in total value, the increasing total value becomes periodic.

We had not expected this at all; but there it was in the data.

What exactly did we have there? Once we started looking, an answer was not hard to find.

We soon realized that the period that we saw in the periodic part of the graph was always exactly the same as the width of one or the other of the small rolls that we were cutting from the wide roll. In fact we found that the small roll that determined the periodicity was always the roll w_j that had the most value per unit of width, the roll that was the most rewarding for the space it used up.

It is plausible that if the size W of the knapsack is big enough, the optimizing solution is likely to have some of these value densest rolls in it. If it does, it is likely that the optimizing solution for a knapsack problem of size $W+w_i$, obtained by adding one more value dense roll to the optimizing solution for the knapsack of size W, has a good chance to be the optimal solution. If this actually occurred for all sufficiently large knapsacks, this behavior would lead to periodicity.

Paul Gilmore and I gave a rigorous explanation along these lines in our paper on the Theory and Computation of Knapsack Functions (Gilmore & Gomory, 1966).

However I had a feeling that there was something more there to be discovered.

[3] An example of that relation for a real problem appears as an appendix in (Gilmore & Gomory, 1963).

4.2. Examining the knapsack: leaving paper trim behind

Now we are going to shift gears and focus on the knapsack problem. The knapsack problem, which is the one-dimensional integer programming problem, appeared in Section 2.6 because of its role in the paper trim problem. Now we focus on it alone and try to better understand its periodicity.

We will also make a transition in notation as we leave the paper trim problem behind.

When working on the paper trim problem the size of the roll that was to be cut up was determined by the width of the paper machine. It was natural to use W for that width and w_j width of the rolls cut out of W. However when talking about the knapsack problem alone, without the paper mill setting it has been, oddly enough, more usual to talk about the length L of the knapsack and lower case letters l_i for the lengths of the individual pieces, and also values v_i rather than prices p_i for the values of the individual lengths. This is what we did, for example, in our paper on the theory and computation of knapsack functions. We will make that notation transition here.

Using length terminology for our knapsack problem we assume we have available n kinds of pieces, the ith kind has length l_i and value v_i. We want to find the integer quantities x_i of each piece that fit in the available length L and have the largest total value V.

We could write the condition of fitting in as an inequality, as we did with the equations in Section 2.2, but here we will write the knapsack problem in equation form. Our pieces will always include one piece of length 1 unit[4] with value 0. Multiples of this piece substitute for an inequality and represents wasted length, and all our other lengths are integer multiples of its length.

So we will move forward with knapsack problems in equation form:

$$\text{maximize} \quad V = \sum_{i=1}^{i=n} v_i x_i \quad \text{with} \quad \sum_{i=1}^{i=n} l_i x_i = L \tag{2}$$

All x_i integer.

5. The relaxed knapsack problem

5.1. Relaxing the knapsack

We start our investigation into periodicity with the linear programming solution to (2), then gradually turn that into an integer solution to the knapsack problem. We will see what this approach can tell us about the periodicity of large knapsack problems and we will illustrate what is happening with a small numerical example.

Starting with the linear programming solution to (2) means that we will fill the entire length L of the knapsack using only the densest piece. We will let x_1 represent the (usually non-integer) quantity of that densest piece, so the basic solution is:

$$x_1 = \frac{L}{l_1} - \sum_{i=2}^{i=n} \frac{l_i}{l_1} x_i \ . \tag{3A}$$

In (3A) the basic variable x_1, which tells us how many rolls of the densest piece are used, has value L/l_1. The non-basic variables x_i are zero. At this basis, the objective function V of (2) looks like this:

$$\text{maximize} \quad V = v_1 \frac{L}{l_1} + \sum_{i=2}^{i=n} v_i^- x_i \quad \text{with all} \quad v_i^- = l_i\left(\frac{v_i}{l_i} - \frac{v_1}{l_1}\right). \tag{3B}$$

[4] The unit is whatever fineness the pieces are being measured to. It could be an inch, a tenth of an inch, or one millimeter

At our current basic feasible solution the objective function V has value $v_1 L / l_1$ and the coefficients $v^-{}_i$ in the objective function row are now all negative. That is as it should be since using any of the non-basic variables takes up space in the knapsack that could be used better by the densest piece x_1.

To find an integer solution to (3A) requires increasing some of the non-basic variables x_i from their present zero values to new positive integer values that make the basic variable x_1 a non-negative integer. We want to find the change in the non-basic variables that produces that x_1 with the least possible increase in cost. Those values of the non-basic variables, together with the x_1 they produce, solve the integer knapsack problem.

Our approach, which will turn out to be the beginning of Corner Polyhedra, will be this: we will focus on finding the least cost change in the non-basic variables that makes x_1 an integer. We will not require x_1 to be non-negative.

Once we know those non-basic variable values, it will turn out to be straightforward to see for what values of L, the knapsack length, the x_1 is non-negative.

5.2. Looking for integer x, the Relaxed Knapsack Equation

Since the condition of being an integer is that $x_1 \equiv 0$, (Mod 1), we apply that condition to (3A). (3A) then becomes what we will call the Relaxed Knapsack Equation.

$$\sum_{i=2}^{i=n} \frac{l_i}{l_1} x_i \equiv \frac{L}{l_1} Mod(1), \tag{4}$$

or equivalently

$$\sum_{i=2}^{i=n} l_i \ x_i \equiv L \, Mod(l_1) \tag{4A}$$

We also make a small change in our objective function (3B), omitting the constant term $v_1 L / l_1$. This gives us an objective function that measures directly the cost of departing from the linear programming solution. So we take as the objective function for the Relaxed Knapsack Equation

$$\text{minimize} \quad V^- = \sum_{i=2}^{i=n} -v_i^- x_i \ . \tag{4B}$$

V^- and all the individual terms on the right are now non-negative.

5.2. The group minimization problem

If we look closely at our new minimization problem, the objective function V^- from (4B) with the Relaxed Knapsack constraint (4A), we will realize that we are really minimizing V^{--}over a group. What exactly does that mean?

Consider the group of the integers (Mod l_1). We can map the integer lengths l_i and L that appear in (4A) into the group elements $l^*{}_i$ and L^* they correspond to in that group. This mapping is addition preserving, so (4A) becomes

$$\sum_{i=2}^{i=n} l_i^* \ x_i = L^* \tag{4C}$$

Conversely if we have group elements $l^*{}_i$ and L^* that satisfy (4C) then if we substitute for those group elements any lengths l_i and L that are mapped into them these lengths will satisfy (4A).

To see concretely what all this means let us look at a small example in a way that allows us to visualize and solve the group minimization problem that we now have.

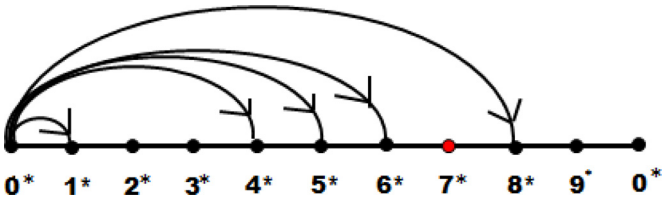

Fig. 2. The group graph H.

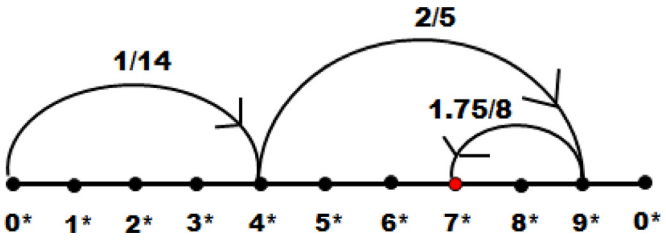

Fig. 3. The minimal cost path to 7*.

5.3. An example and the group graph H

Here is a very small example. We have a knapsack of length 27 and pieces of lengths l_i = 10, 14, 8, 6, 5 and 1. The values v_i of the pieces are 15, 20, 10.25, 5.5, 5.5 and 0. The piece of most value per unit length is the piece of length 10.

In (4A) we replace lengths l_i by their corresponding group elements from the group of integers (Mod 10). Using * to indicate group elements, our pieces of lengths 1, 5, 6, 8, and 14 correspond to the group elements 1*, 5*, 6*, 8*, and 4*. The knapsack length 27 corresponds to the group element 7*. If the knapsack length L had been 17 or 37 or 47, rather than 27, L* would still be is 7*. However if L had been 28, L* would be 8*. All possible knapsack lengths end up being one of the 10 group elements.

We will call L^* the group goal element.

Fig. 2 shows the integers (Mod 10) as nodes of a graph which we will refer to as the group graph H. The arcs in the group graph show the effect of adding to the group element 0* the group elements (1*, 4*, 5*, 6*, 8*), which correspond to the knapsack pieces of length (1, 14, 5, 6, 8).

The goal element L^* is 7* and is shown in red.

The connection of the group graph with finding solutions to (4A) is very direct. Any path p in H leading from 0* to the goal element L* produces a solution to (4A). The values of l_i and L that solve 4A are any values that map into the group elements included in the path p, and the x_i that appear in (4A) are the number of times the ith group element appears in the path.

5.4. Least cost paths in H

If we assign to the ith arc, as its cost, the value $-v^{--}{}_{i}$, the minimization problem (4B) becomes the problem of finding the least cost path from 0* to L^* in the graph H based on the costs $-v^{--}{}_{i}$.

In the path shown in Fig. 3 the first number on each arc is $v^{-}{}_{i}$, while the second number is the actual length of the piece.

The minimal cost path from 0* to 7* is quickly found[5] to be {4*, 5*, 8*}, the path shown in Fig. 3. The total cost of the path is 1+2+1.75 = 4.75 and its total length is 14+5+8 = 27.

For knapsacks of any length L equivalent to 7, the total path cost 4.75 is the reduction in the value V caused by raising the non-basic variables attached to the pieces of length 14, 5, and 8 from 0 to 1.

[5] Any shortest path method, including dynamic programming, will do. Methods for solving knapsack problems are discussed extensively in (Gilmore & Gomory, 1966).

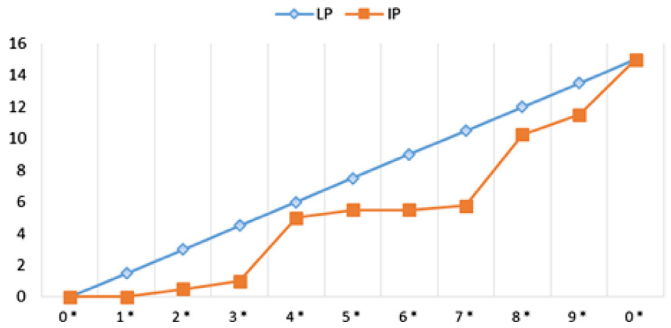

Fig. 4. The cost of being integer.

This change will produce an integer x_1 for any knapsack with L equivalent to 7 (Mod 10) so it applies to knapsacks of length L = 7, 17, 27, 37, 47, 57… and so on indefinitely.

We can follow this same procedure if the goal element is not 7* but instead any other group element. We will find that the cost of the least cost paths from 0* to goal elements g = (1*, 2*, 3*, 4*, 5*, 6*, 7*, 8*, 9*) are (1.5, 2.75, 3.75, 1, 2, 3.5, 4.75, 1.75, 3) respectively, and the total length of each of those least cost paths is TL(g) = (1, 22, 13, 14, 5, 6, 27, 8, 19) .

In Fig. 4 the vertical axis displays the increases in value as the knapsack length is increased by 10. The horizontal axis shows the group goal element. The blue line shows the increase in the L.P. solution value. The red line shows the increase in the value obtained with the best integer solution.

This periodic pattern repeats indefinitely.

Now the question is: when are the integers x_1 that are being produced by this periodic pattern non-negative integers? When they are non-negative they solve the original knapsack problem.

5.6. Critical length for non-negativity

Fortunately it is now straightforward to find out what values of the knapsack length L make x_1 non-negative.

We will ask and answer that question for our example, then the method will be plain. We have goal element 7* and therefore our least cost path is, as shown in Section 5.4, is (4*, 5*, 8*). These are the non-basics that were 0 in the LP solution but are now 1 in the least cost path. To determine x_1 we substitute that path value into the sum in (3A). Putting the lengths (14, 5, 8) in the sum and multiplying through by l_1 leaves us with $10x_1$ = L-28.

Now the situation is clear, for goal element 7* we will have x_1 values that are negative if $L < 28$, for example 7 or 17 or 27. This does not provide a solution to the knapsack problem. If L is 28, x_1 will be 0, a valid solution, there will be no rolls of 10 in the knapsack solution for this special value. For L = 37, there will be one densest roll, for L = 47 there will be two of them, and then the solution to the knapsack problem continues indefinably adding more and more of the densest roll to the fixed values of the non-basic variables that appear in the least cost path.

More generally, given a goal element and its attendant least cost path we substitute its x_i values and their lengths in the sum in (3A) which is simply the total path length TL(g^*) of the least cost path.

We then obtain $l_1\ x_1 = L$–TL(g^*). So the critical value is L = T(g^*). For $L \geq$ T(g^*) x_1 will be non-negative and solves the knapsack problem for goal element g^*.

To find solutions for the whole knapsack problem we simply find the group element g^* with largest TL(g^*) value TLmax. For $L \geq$ TLmax our least cost paths solve the Knapsack problem; we simply use the least cost paths with increasing numbers of densest rolls added.

This provides periodicity with period l_1.

For problems with goal group elements (0*,1*, 2*, 3*, 4*, 5*, 6*, 7*, 8*, 9*), the least cost path lengths turn out to be (0, 1, 22, 13, 14, 5, 6, 27, 8, 19).

Therefore for knapsack lengths of 27 or longer, all the group shortest paths can be used for the knapsack solution with the appropriate number of 10's added. For all L we will have complete periodicity of order 10 in the knapsack.

5.4. Summary

We can follow this procedure for any integer knapsack problem. We first select the piece of length l_i having greatest value density. We form the group graph of all the corresponding elements l_i* where the * indicates (Mod l_1). In the group graph we find the shortest path from 0* to each group element using the values $-v_i$*. Each of these shortest paths will then represent the special solution for knapsack lengths that are equivalent (Mod l_1) to that group element.

Each of those shortest paths will have an actual length using the lengths of the original pieces. If the longest of these path lengths is L_{max}, then for any $L \geq T$ Lmax these paths, with densest pieces l_1 added, solve the original knapsack problem and produce periodicity.

Introducing the group relaxation has explained the source of the periodicity we first observed in the actual data and given us solutions for large knapsack problems or for integer programming problems of one dimension. We will next explore Corner Polyhedra for integer programming problems of more than one dimension

6. General integer programming and the group relaxation [6]

6.1. The group equations and corner polyhedra

Now we apply the same relaxation to general integer programs. It is the solutions to the relaxed problem that produce Corner Polyhedra.

We will see that the relaxed problem is very structured and lends itself to theoretical analysis surprisingly well. We will also see cutting planes for the original problem emerge from knowledge gained about the relaxed problem.

The general integer programming problem can be stated as maximizing c · x over the polyhedron in n-space

$$Ax = Bx_B + Nx_N = b, \quad x \geq 0 \ \ and\, x \ \ integer \tag{5}$$

Here A is an $(m+n)$ x m matrix which divides into an m x m matrix B and an m x n matrix N while the $m+n$ integer vector x divides into an m vector x_B and an n vector x_N.

We choose B as our basis and solve for the resulting basic variables x_B.

$$Ix_B + B^{-1}Nx_N = B^{-1}b \tag{5A}$$

Just as in the knapsack case we relax this to a group equation by dropping the non-negativity requirement on x_B and requiring only that x_B be an integer vector. Any x_B from (5A) will be integer if x_N satisfies what are now group equations:[7]

$$B^{-1}Nx_N \equiv B^{-1}b \quad (Mod\ 1) \tag{6A}$$

or equivalently

$$Nx_N \equiv b \quad (Mod\, B) \tag{6B}$$

Just as in our knapsack example, we are now dealing only with the non-basic variables, they must be chosen in (6A) or (6B) to make the x_B integer.

We can rewrite (6A) and (6B) to make the individual columns and variables more visible. We write $c^N{}_j$ for the columns of N, and use $c^T{}_j$ for the columns of the transformed N matrix $B^{-1}N$. The individual non-basic variables that accompany those columns we denote by $x^N{}_j$. Using this notation the equivalent Eqs. 6(A) and (6B) become the equivalent equations

$$\sum_j x_j^N c_j^T \equiv b^T \quad (Mod\ 1) \tag{6C}$$

or equivalently

$$\sum_j x_j^N c_j^N \equiv \mathrm{b} \quad (Mod\ \mathrm{B}) \tag{6D}$$

Where $b^T = B^{-1}b$.

These are group equations, because in (6C) it makes no difference if we replace the columns $c^T{}_j$ or the right hand side element b^T with a different column that is equivalent to it (Mod 1), or in (6D) if we replace any of the $c^N{}_j$ or b with columns equivalent to them (Mod B).

We will discuss the properties of our new group Eqs. (6C) and (6D) as we go along. But for the moment we will leave Algebra and introduce Geometry.

6.2. Geometry of corner polyhedra; cutting planes

Fig. 5 shows here the polyhedron of a standard linear program.

Each thin black line in Fig. 5 indicates an inequality. The dots are the integer values of the variables. At the vertex V, associated with the basis B, we have a basic feasible solution. At V the basic variables x_B are non-negative and the non-basic variables x_N are zero.

The dark gray area in the Fig. 5 displays the integer programming (IP) polyhedron. It is the convex hull of the feasible solutions that are integer; so in Fig. 5 it is the convex hull of the dots lying inside the linear programming polyhedron.

When we substitute for the Integer Programming Eq. (5A) the group Eqs. (6A) or (6B) we are getting integer solutions at the price of giving up the non-negativity of the basic variables. Since we are

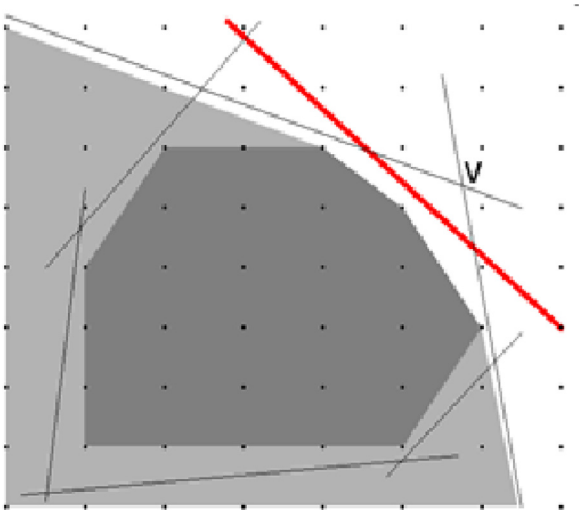

Fig. 5. LP, IP, and corner polyhedra.

[6] The basic reference for all of this is (Gomory, 1969). The language there is antiquated, the special solution there is called asymptotic integer programming, but the content is the same.

[7] A vector v_1 in n-space is equivalent to another vector v_2, written $v_1 \equiv v_2$ (Mod B), and represents the same group element (Mod B) if v_2 can be reached from v_1 by adding to v_1 an integer sum of the columns of B. Given the matrix B the group of the integers can be easily found by a standard matrix calculation. One is available, for example in MatLab under the heading smithnormalDform.

now disregarding these inequalities, we have additional integer solutions that lie outside the linear programming polyhedron. The convex hull of all the integer solutions to the relaxed problem (6A) or (6B) is the Corner Polyhedron.

In Fig. 5 the dots in the integer programming polyhedron have been joined by additional dots that lie in the light gray area. The dark and light areas together make up the Corner Polyhedron.

Fig. 5 makes one important point very clear: since the integer programming region lies inside the Corner Polyhedron, *cutting planes for Corner Polyhedra,* such as the red line in Fig. 5, *are always cutting planes for the integer problem*. Therefore in our examination of Corner Polyhedra we will keep cutting planes very much in mind.

Fig. 5 also suggests that in some cases the Corner Polyhedron might be very similar to the Integer Polyhedron in the region near the linear programming vertex.

6.3. N-space

Learning about the geometry of Corner Polyhedra is facilitated by introducing N-Space. Imagine that you are standing on the linear programming vertex V of Fig. 5 looking in at the corner polyhedron. What you would see is something like this:

The vertex V, where all components of x_N are zero, is now the origin; everything is located by specifying the values x_N . We will refer to this coordinate system as N-Space. In Fig. 6 the small black dots are points where x_N is an integer vector. However, unlike Fig. 5, where all the dots represented integer x_N and x_B, the dots in Fig. 6 indicate integer x_N but it is only the circled dots that satisfy the group equation and produce integer x_B.

We also show cutting planes In Fig. 6. Cutting planes for corner polyhedral are inequalities (π, π_0) in *N* space that have the origin on one side and all the circled dots, all the solutions to the group equation, on the other.

Cutting planes for Corner Polyhedra come in all degrees of strength, but actual facets of the Corner Polyhedron would certainly make the strongest cutting planes. So it would help, if we could, to learn something about the facets of Corner Polyhedra.

Surprisingly, a great deal can be learned, much more than I expected when I started down this path. But to get to these results we need put our focus squarely on the groups and the group Eqs. (6A) and (6B) rather than on the particular integer program that gave rise to them.

6.4. Separating corner polyhedra from their I.P. problems

So far our Corner Polyhedra have been tightly tied to the B, N, and b of the integer programming problem. It is natural therefore to adopt the notation P{B, N, b) for the Corner Polyhedron whose group G is the integers (Mod B), whose list of group elements consists of the columns c_j^N of N (Mod B), and whose goal element g_0 is b (Mod B).

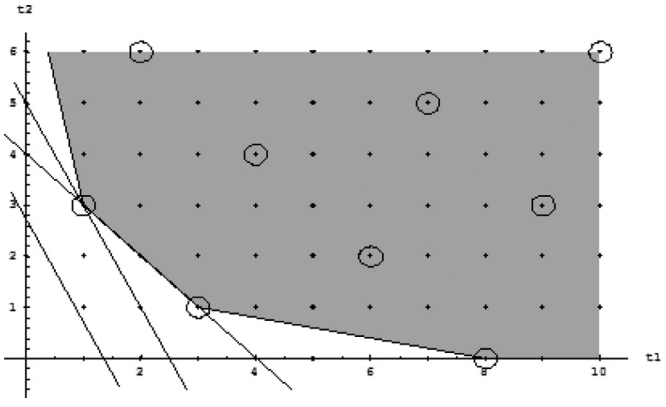

Fig. 6. N-space. The corner polyhedron viewed from the LP vertex V.

However the same group G, group elements {g}, and goal element g_0 can be produced by many different B, N, and b.[8] So, for the moment, we are going to step away from B, N and b and the integer programming origins of corner polyhedra and take the G, {g}, g_0 and the corner polyhedron they generate, as objects of study in themselves.

When we know more about these polyhedra we will return and use that knowledge on the original Integer programming problem. At that point we will have very direct and simple process that enables us to generate whole new families of cutting planes directly from the usual linear programming data and without any discussion of groups and Corner Polyhedra. That will be the content of Section 9.4.

In this next section we start down the path that leads to a considerable understanding of Corner Polyhedra and eventually to the process of Section 9.4

7. Corner polyhedra of groups

7.1. Defining the group polyhedra: N space, T-space, and the group diagram

P(B, N, b) is the convex hull in N space of the non-basic variables x_N that satisfy

$$\sum_j x_j^N c_j^N \equiv b \quad (Mod\ B). \qquad (6D)$$

Let us take as our group G, the integers (Mod B), and take for our group elements g the elements of G that correspond to the columns $c^N{}_j$, and take the group element that corresponds to b as the goal element g_0. We define the Corner Polyhedron P(G, {g}, g_0) for the group G, the elements list {g} and goal element g_0 as the convex hull in T-space of the non-negative integer points t(g) that satisfy the Group Equation that corresponds directly to (6D).

$$\sum_{g\in\{g\}} t(g)g = g_0 \qquad (7)$$

Instead of the columns of N adding up to b Mod (B), as in (6D), we have in (7) the corresponding group elements forming a path from 0^* to g_0 in G. In T-space the group elements g play the role of the columns of N in (6D), and the *t* (g) play the role of the non-basic variables x^N.

T space and N space are almost exactly the same. However in T-Space we deal with the underlying group G and its elements not with the matrix B and its columns which is one of many that give rise to the same G and {g}.

The integer points in T-Space that satisfy (7) are the ones we are interested in. The convex hull of these points form the Corner Polyhedron in T space and each one of the points represents a path in the group from the zero element of G to the goal element g_0. We will refer to the collection of these points in T-Space as the set T^*.

We start by defining cutting planes for Group Corner Polyhedra.

[8] We would describe our Knapsack example with L = 17 as P((10), {1, 5, 6, 8, 14}, 17). The group description P(G, {g}, g_0) would be P((10), {1^*, 4^*, 5^*, 6^*, 8^*}, 7^*) which could also be produced by P((10), {1, 5, 6, 8, 14, 16, 18}, 27). More generally in n-space (Mod B) two column vectors are equivalent if one can be reached from the other by adding columns of B. So adding an integer multiple of one column of B to another column of B produces a B' that is different, but n-space (Mod B') is the same group as n-space (Mod B).

7.2. Cutting planes

A cutting plane for a Group Corner Polyhedron is simply an inequality (π, π_0) that has all the elements of T^* on one side. More precisely, (π, π_0), with non-negative[9] π and π_0, is a cutting plane if and only if:

$$\sum_{g\in\{g\}} \pi(g)t(g) \geq \pi_0 \ \geq 0 \tag{8}$$

for all t(g) satisfying (7). A cutting plane (π, π_0) assigns a path length $\pi \cdot t \geq \pi_0$ to any path from 0^* to the goal element g_0.

Since (8) is just a very long list of inequalities, it has basic feasible solutions.

If we could get basic feasible solutions to (8) what would we have? Since basic feasible solutions to a list of inequalities produce as much equality as is possible, a basic feasible solution would be a cutting plane that is pressed up against the set T^* with as much contact as possible. Such a cutting plane would be a facet of the Group Corner Polyhedron.

More formally:

Facet Theorem[10]. The inequality (π, π_0) provides a facet of P(G, {g}, g_0) if and only if (π, π_0) is a basic feasible solution of the list of inequalities (8).

When I first saw this I was really surprised, I thought it was a remarkable situation. I had always thought it was very hard to find facets of polyhedra that are defined as the convex hull of a set of integer points, but here was a large ordinary linear programming problem whose basic feasible solutions turn out to be the facets of an integer polyhedron, the Corner Polyhedron.

In addition, the large linear programming problem had lots of structure.

As it turned out there was more structure than even I could have hoped. We are going to be able to learn a surprising amount about the structure of those basic feasible solutions, and there are some fundamental facts about group cutting planes that will help us to get there.

7.3. Subadditivity of cutting planes

A cutting plane π (g) is said to have the property of subadditivity if, whenever π (g) is defined on group elements g_1, g_2 and on their sum element g_1+g_2, we always have $\pi(g_1)+\pi(g_2) \geq \pi(g_1+g_2)$. At this moment this may seem like an arbitrary property, but we will see in Section 8.2 that it has enormous consequences.

Fortunately, for a cutting plane, being subadditive is the most natural thing in the world. Suppose we have a cutting plane π (g) that is not subadditive, so it has $\pi(g_1)+\pi(g_2) < \pi(g_1+g_2)$ for some particular g_1 and g_2.

We can change π to a strictly stronger inequality π' by simply lowering the value of $\pi'(g_1+g_2)$ so that $\pi'(g_1+g_2) = \pi(g_1)+\pi(g_2)$, but otherwise letting $\pi'(g) = \pi(g)$. We claim this new stronger inequality is still a cutting plane.

Proof: If the new inequality $\pi'(g)$ fails to be a cutting plane it is because there is some path in T^* from 0^* to g_0 involving the changed element (g_1+g_2) but having total path length $\pi \cdot t < \pi_0$ so that it fails to satisfy (8). But if in that hypothetical path we replace the element g_1+g_2 by the separate elements g_1 and g_2, since $\pi(g_1)$ and $\pi(g_2)$ have not changed, that path would have the same length as it did before the change from π to π'. So it could not be $< \pi_0$. So $\pi'(g)$ is in fact a new and stronger cutting plane.

Since any non-subadditive cutting plane can be strengthened into a cutting plane that is subadditive, from here on we will assume that all our cutting planes are subadditive.

We are now ready to tackle the facets of the Group Corner Polyhedra. We will start with the Master Polyhedra.

8. Facets and master polyhedra

8.1. Master polyhedra: definition and Structure Theorem

If the list of group elements {g} contains all the non-zero elements of G, we call P(G, {g}, g_0) a Master Polyhedron and denote it by P(G, g_0). We will see that P(G, g_0) has a structure which sheds light on the structure of all its large variety of related group polyhedra P(G, {g}, g_0).

We can now state the fundamental Structure Theorem of Master Corner Polyhedra.

Structure Theorem[11]: (π, π_0) is a facet of the Polyhedron P(G, g_0), if and only if π is a basic feasible solution to the system of equations and inequalities:

$$\begin{array}{ll} \pi(g_0) = \pi_0 & \\ \pi(g) + \pi(g_0 - g) = \pi_0 & g \in G^+, \quad g \neq g_0 \\ \pi(g) + \pi(g') \geq \pi + (g+g') & g, g' \in G^+ \\ \pi(g) \geq 0 & g \in G^+ \end{array} \tag{9}$$

I was already surprised when I saw in the Facets Theorem that a large linear program involving all the elements of T^* could give us facets of the Corner Polyhedron. I was even more surprised when I found this Structure Theorem which shows, at least for Master Polyhedra, that we can replace the long list of inequalities of the Facet Theorem (Section 7.2) by a very much shorter and very structured list of equations and inequalities.

When I first saw this theorem, I thought it was too good to be true. Fortunately it really is true. With the benefit of hindsight we can say this: the inequalities in the theorem reflect the property of subadditivity, the equations reflect the property of minimality, or symmetry.

Here is a way of seeing where the Structure Theorem comes from.

8.2. Origin of the Structure Theorem

Consider a cutting plane (π, π_0) we obtain as a basic feasible solution to the long list of inequalities (8) in Section 7.2. According to the Facet Theorem 7.2 (π, π_0) is a facet, there cannot be a stronger inequality, so (π, π_0) must already be subadditive. Choose any row t for which we have equality in (8). The row represents a path from 0^* to g_0. Take any two elements g_1 and g_2 of that path and replace them by the sum element (g_1+g_2). Because we have subadditivity, $\pi(g_1)+\pi(g_2) \geq \pi(g_1+g_2)$, this change must either decrease the total path length $\pi \cdot t$ or leave it unchanged. Since our path already has equality in (8) its length cannot be decreased, so we must have equality: $\pi(g_1)+\pi(g_2) = \pi(g_1+g_2)$.

Now replace the row t with two equalities; the changed row t, which now has one less element in it, and the new equality $\pi(g_1)+\pi(g_2) = \pi(g_1+g_2)$. These two along with the other equalities of the basic feasible solution, still determine the same (π, π_0) because the change is reversible, the two new rows added together reproduce the original path t.

Now we keep repeating this process until all the original rows have been reduced to two elements each and a number of subadditive equalities have been added.[12] But this is then a collection

[9] The non-negativity does not have to be assumed, it is actually a consequence of the structure of the Group Polyhedra. See supplementary material Appendix A.

[10] This is Theorem 7 of (Gomory, 1969) but first appeared in (Gomory, 1967).

[11] This is Theorem 18 of (Gomory, 1969). For simplicity the notes given there about special cases have been omitted here.

[12] There will be considerable duplication between the new rows and already existing rows as we go through this process.

of equalities from the Structure Theorem array (9) and that collection of equalities still determine the same (π, π_0). So we have produced a basic feasible solution of (9) that matches the one we started from in (8).

In addition, since the whole process is reversible, we can start with a basic feasible solution to (9) and produce a corresponding one in (8) that determines the same facet (π, π_0).

We can make this surprising result more concrete by looking at our little example.

8.3. The master polyhedron for the small example

Here are the inequalities that give us the facets of the integer polyhedron $P(G_{10}, 7^*)$ corresponding to our small example. In (π, π_0) we choose $\pi_0 = 1$

$$\pi_1 + \pi_6 = 1 \quad \pi_2 + \pi_5 = 1 \quad \pi_3 + \pi_4 = 1 \quad \pi_8 + \pi_9 = 1$$
$$\pi_1 + \pi_1 \geq \pi_2 \quad \pi_2 + \pi_1 \geq \pi_3 \pi_3 + \pi_1 \geq \pi_4$$
$$\pi_8 + \pi_8 \geq \pi_6 \quad \pi_9 + \pi_9 \geq \pi_8$$

There are four equalities, and then 29 inequalities of which only five are shown here explicitly. But all 29 are about subadditivity; they all say that for any pair of group elements if you add their π values together, the result is at least as large as the π value of the resulting group element.

So what you have in our small example is a sparse matrix of 0's and 1's where the position of the 1's reflects the group structure.

It is clear that the matrix will grow rapidly in size but there it is, it is reality. If it grows it is because that is the way that Corner Polyhedra, and the integer programming problems they are related to, actually are.

8.4. Using the Structure Theorem – seeing structured facets,

It was very exciting for me to use this theorem to actually compute the Master Polyhedra for various groups. I was able to use a program of Balinsky and Wolfe (Wolfe, 1963) that finds all the vertices and faces of a system of linear inequalities. Using the Balinski–Wolfe program I saw for first time the complete structure of a corner polyhedron, or for that matter of any integer polyhedron. Although I had originated the first general cutting planes for IP (Gomory, 1958), and solved many IP problems, I had never before seen all the facets of any but the very tiniest polyhedra that were convex hulls of integer points.

But now I could see them. The Balinsky–Wolfe program enabled me to find all the facets and vertices for all groups of 12 or less elements. They are all listed in the appendix of (Gomory, 1969).

Row	π 1*	π 2	π 3	π 4*	π 5*	π 6*	π 7	π 8*	π 9	π 0
1	1	0	1	0	1	0	1	0	1	1
2	1	2	1	2	1	2	3	2	1	3
3	2	4	1	3	0	2	4	1	3	4
4	3	6	4	2	0	3	6	4	2	6
5	3	1	4	2	5	3	6	4	2	6
6	4	3	2	6	5	4	8	2	6	8
7	2	4	6	3	5	7	9	6	3	9
8	7	4	6	3	5	2	9	6	3	9
9	7	4	1	8	5	2	9	6	3	9
10	3	6	9	2	5	8	11	4	7	11
11	2	4	6	8	10	12	14	6	8	14
12	3	6	9	12	15	18	21	14	7	21

Fig. 7. Facets of $P(G_{10}, 7^*)$.

Based on that appendix, Fig. 7. lists all of the facets of the Master Polyhedron $P(G_{10}, 7^*)$ of our example.[13] In this table we have not listed the non-negativity constraints on the variables which are always facets except for the variable corresponding to the goal element.

Now let's look at these facets. Row 1 certainly looks structured. Row 12 looks structured too. In fact it looks suspiciously like the Gomory Mixed Integer Cut (GMIC) which we will discuss in Section 9.1. If we look at row after row almost all look anything but random.

The presence of this structure in the facets is very encouraging and we will return to this topic in Section 9.1. However we should first ask if our understanding of Master Polyhedra connects with the need to analyze non-master polyhedra. Our knapsack example $P(G_{10}, \{1^*, 4^*, 5^*, 6^*, 8^*\}, 7^*)$ was certainly not a master polyhedron, it did not have all the group elements present.

8.5. Returning to the non-master polyhedra: The Intersection Theorem

The Intersection Theorem[14] describes the relation between the Master Polyhedron $P(G, g_0)$ their many possible related $P(G, \{g\}, g_0)$.

Intersection Theorem:

$$P(G, \{g\}, g_0) = P(G, g_0) \cap^E (\{g\})$$

$E\{g\}$ refers to the subspace of T-Space generated by the elements of {g}.

The theorem asserts that $P(G, \{g\}, g_0)$ is simply the part of $P(G, g_0)$ that lies in the subspace generated by the elements of {g}.

For vertices it means that the vertices of $P(G,\{g\}, g_0)$ are obtained by taking the vertices of $P(G, g_0)$ and eliminating in each of them any positive terms referring to g that are not in {g}.

For facets it is more complicated. It is true that if we take all the facets of $P(G, g_0)$, and eliminate the terms not in {g}, the intersection of the resulting inequalities is $P(G, \{g\}, g_0)$. However this is not the same as saying that all these inequalities are facets of $P(G, \{g\}, g_0)$. Some will be facets and some will be valid inequalities but not facets. The list of these inequalities will contain every single facet of $P(G, \{g\}, g_0)$, but in addition there will be many others that are valid inequalities for $P(G, \{g\}, g_0)$, but, since we already have all the facets of $P(G, g_0)$, are now superfluous.

However I was pleasantly surprised to get this much. Generally you can't intersect the convex hull of a set of integer points with the convex hull of another set and get the convex hull of the integer points they have in common, but in this case we can.

Fig. 8 shows our small problem. Since we don't have a 9^* or a 7^*, a 3^* or a 2^* in our problem, we just strike out those columns and the resulting inequalities define the corner polyhedron $P(G_{10}, \{1^*, 4^*, 5^*, 6^*, 8^*\}, 7^*)$.

If we want to know if a particular row is a facet, we could check to see how many of our vertices lie on it. I have done that for goal element 7^* and the result is that rows 1, 3, 4, 10 and 11 are facets.

We can also check which of these facets are incident to the vertex $(4^*, 5^*, 8^*)$ which was the solution to our original problem knapsack problem for $L \geq 28$. These are rows 1, 4, and 10. These, together with the non-negativity constraints give us a complete picture of the polyhedron in the neighborhood of the maximizing vertex.

[13] The appendix does not list the facets of $P(G_{10}, g_0)$ for all g_0, it only provides all the automorphism classes, i.e. the missing g_0 can be obtained from the ones that are listed by simply multiplying one of the listed g_0 by an integer. In our case $P(G_{10}, 9^*)$ is listed so we get the facets of $P(g_0, 7^*)$ by multiplying all the group elements by 3 which sends 9 into 7 Mod(10).

[14] Gomory (1969, Theorem 12).

Row	π 1*	π 4*	π 5*	π 6*	π 8*	π 0
1	1	0	1	0	0	1
2	1	2	1	2	2	3
3	2	3	0	2	1	4
4	3	2	0	3	4	6
5	3	2	5	3	4	6
6	4	6	5	4	2	8
7	2	3	5	7	6	9
8	7	3	5	2	6	9
9	7	8	5	2	6	9
10	3	2	5	8	4	11
11	2	8	10	12	6	14
12	3	12	15	18	14	21

Fig. 8. Inequalities defining the corner polyhedron $P(G_{10}, \{1^*, 4^*, 5^*, 6^*, 8^*\}, 7^*)$.

8.6. Learning more: shooting theorem and experiments[15]

The presence of all this structure is very encouraging; but you quickly find that you would like to learn more and deal with larger problems. However the work of finding all the vertices of a polyhedron grows rapidly with problem size.

But there are some interesting questions that can be attacked without finding all those vertices and facets. Here is one: are there a few big important facets and a lot of little ones, or is there a fairly even spread of sizes? If there are big important facets, are they the very structured ones?

I always thought it would be wonderful if you could shoot arrows at the corner polyhedron from the origin and find out where they hit. If you fired randomly you would hit big faces often and very little ones very rarely. But it seems that you would have to compute those faces before you could shoot at them, and that computation is just what we are trying to avoid.

What is remarkable is that there is a way to do this and I managed not to see it for many years. Here is the Shooting Theorem that I first described in a lecture at Georgia Tech in 1998.

Consider a vector v that points into the first quadrant. We can think of this as a shooting direction and the direction can be randomly chosen (see Fig. 9). The following theorem, based on the Structure Theorem constraints (9) in Section 8.1, tells us which facet would be hit if we fired in the direction v.

[15] The work on shooting was not done until much later but it makes a more coherent picture to discuss it here.

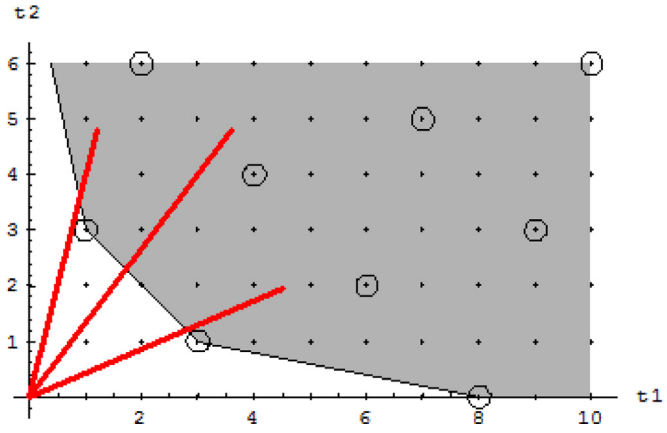

Fig. 9. The shooting experiment.

Shooting Theorem: The facet π that solves the problem of minimizing πv subject to the constraints:

$$\begin{array}{ll} \text{Min } \pi v & \\ \pi(g) + \pi(g_0 - g) = \pi_0 & g \in G^+, \quad g \neq g_0 \\ \pi(g) + \pi(g') \geq \pi(g + g') & g, g' \in G^+ \\ \pi(g_0) = \pi_0\ , \quad \pi(g) \geq 0 & g \in G^+ \end{array} \tag{9a}$$

is the facet first hit by the direction v.

What is remarkable is that we are finding the facet that v hits, and doing that without any list of facets. We just use the constraints of the Structure Theorem, and the right facet pops out.

8.7. What we found out by shooting

After I had done some very preliminary shooting calculations, Ellis Johnson and Lisa Evans took up the challenge of doing full scale computations, firing 10,000 shots each at master polyhedra of sizes up to 20. The results which we described in a joint paper (Gomory, Johnson, & Evans, 2003) I still find very striking (Fig. 10).

Especially significant is the line in Fig. 10 entitled "Facets Hit to 50%". For example in shooting at the Polyhedron $P(G_{18}, 2)$, the 10,000 shots were spread over only 151 facets. But, even more concentrated, 50% or 5, 000 of those hits were on only 8 of the facets. Not only was there a great concentration of hits on these 8 facets, but in looking at these facets there was a remarkable dominance of very simple facets. Many showed the role of automorphisms and of subgroups.

Other researchers have since done much larger and more decisive experiments that confirm the dominance we see here of a small number of large structured facets.

At this point it is natural to try to connect what we have learned about Group Polyhedra to the integer programming problems of the real world.

9. Large problems

9.1. Structured facets

One way to make progress with large problems is to look at the structured facets that show up in small examples, or are hit by shooting, and see if the structure can be carried up to large problems. Here is one example:

Look at our familiar cyclic group of order 10 with goal element 7*. We are interested in facets (π, π_0) and choose $\pi_0 = 1$. Then, for all the group elements taken in order, 1*, 2*, 3*,..., choose the values $\pi = \{1/7, 2/7, 3/7, 4/7, 5/7, 6/7, 1, 2/3, 1/3\}$. Multiplied through by 21 to make these into integers, this is the GMIC and also the last row in our list of facets in Fig. 7.

This GMIC facet structure generalizes to any cyclic group G no matter how large and any goal element g_0.

Assume that the goal element is the rth element of the cyclic group. We choose value 1 for it. For the ith element, if $i \leq r$, we assign the value $\pi(g_i) = i/r$, for $i > r$, we choose $\pi(g_i) = (|G| - i)/(|G| - r)$.[16] This is a facet for any cyclic group and any goal element.

Now consider automorphisms. If the goal element in our example had been 9* instead of 7* we would have had for G(10, 9*) the facet $\pi = (1/9, 2/9, 3/9, 4/9, 5/9, 7/9, 8/9, 1\}$ or in integers (1, 2, 3, 4, 5 6 7, 8, 9}. Multiplying the elements of our cyclic group of 10 by 3 is an automorphism that sends element 9* onto 7*; sure enough, the numbers 1–9, properly rearranged, show up as a facet in the fourth row from the bottom in Fig. 7.

Automorphisms of Master Polyhedra map facets into facets. It is also true that facets of subgroups lift up and reappear as facets

[16] This is the integer part of the Gomory mixed integer cut.

Polyhedron	$P(G_{18}, 2)$	$P(G_{18}, 3)$	$P(G_{18}, 6)$	$P(G_{18}, 9)$
Facets Hit in 10,000	151	479	207	505
Facets Hit to 50%	8	11	17	12

Polyhedron	$P(G_{20},2)$	$P(G_{20},4\}$	$P(G_{20},5\}$	$P(G_{20},10\}$
Facets Hit in 10,000	341	587	929	371
Facets Hit to 50%	10	19	20	25

Fig. 10. Shooting results.

for their parent groups. The same GMIC construction used on a subgroup of order 5 reappears twice in row 3 of Fig. 7 to form a facet of our group of order 10.

When it comes to exploiting structures, the effect of automorphisms and lifting up from subgroups, there is an almost endless list of possibilities. Sometimes these explorations can be difficult; sometimes the automorphism of a very symmetric looking facet can look quite jumbled. This section barely touches on what is possible.

However dealing with arbitrarily large cyclic groups takes us naturally to our next step. We move from finite cyclic groups to the group formed by the real numbers (Mod1).

9.2. The real line (Mod 1)

There is not much difference between the reals modulo 1, which we will refer to as RM1, and any very large cyclic group. However in RM1 we do have the freedom to take as group elements any numbers in the interval [0, 1) and add them Mod 1 to produce another group element. We don't need to pick elements from some fixed grid of points. This will open the door to the consideration of continuous variables along with integer ones. This is work that Ellis Johnson and I did together in the early 1970's (Gomory & Johnson, 1972; Gomory & Johnson, 1972; Gomory & Johnson, 1973).

Working in RM1 we can define a path to a goal element as in Eq. (7), and we can define cutting planes as in Eq. (8). However in our work with RM1, which contains infinity of group members, we will confine the sums in these equations to adding up a finite number of group elements.[17] Any one path, for example, is allowed to contain only a finite number of group elements.

We will still want to distinguish facets from lesser cutting planes. A good definition for a facet, that includes what we have done so far with basic feasible solutions, is this: a cutting plane (π, π_0) is a facet if there exists no other cutting plane (π', π_0) such that $\pi'(g) \leq \pi(g)$ for all g, and also for at least one g we have strict inequality, $\pi'(g) < \pi(g)$.

Fig. 11 shows an example of a cutting plane for RM1. This is the direct extension to RM1 of the GMIC discussed in Section 9.1. We have simply done straight line interpolation between the elements of the cyclic group.

We are fortunate that the most obvious methods for extending our finite group facets to RM1 actually work. In addition to straight line interpolation, Ellis Johnson and I included in our work a discussion of a different method of interpolation involving connecting points using two slopes rather than straight lines. That concept that will seem more natural after reading Section 9.6;

9.3. Moving to large problems: not knowing G

We are now ready to move on to problems of any size. For really large problems it would be great to have the ability to generate cutting planes without even having to find out what the group G determined by the basis B actually is.

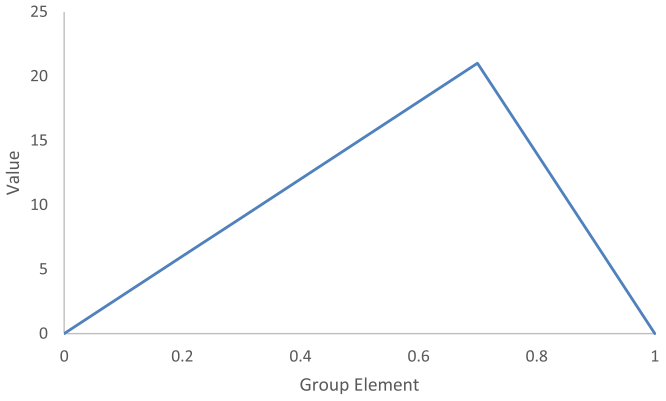

Fig. 11. A cutting plane for RM1.

Although this sounds like a lot to ask, we might suspect that it is possible; the original Gomory Fractional Cuts were generated by taking the fractional parts of the entries in rows of the matrix $B^{-1}N$ that appears in Eq. (6A) and using them as cutting planes. There was no need to know what group was being produced by the matrix B. Can we do something like that here?

Fortunately, with what we know now, we are able to produce whole new families of inequalities for Corner Polyhedra without analyzing B. Our basic approach[18], described in detail in the next section, is based on the fractional rows used in the original Gomory Fractional Cuts but here we use the fractional rows not as cuts but as mappings from the columns of the Corner Polyhedron, as displayed in Eq. (6C), into group elements in RM1. We will then be able to carry back cutting planes from the group RM1 to form cutting planes for the Corner Polyhedron.

9.4. How cutting planes for corner polyhedra are produced from group cutting planes

Eq. (6C) displays the columns of $B^{-1}N$ as the c^T_j alongside their accompanying non-basic variables x^N_j. We take the ith row of Eq. (6C), map it into a path in the real line (Mod 1) by replacing each row element $c_{i,j}$ by the group element g(j) it corresponds to (Mod 1). For example if $c_{i,j}$ was −7.61, the corresponding g(j) would be 0.39*.

We also replace the right hand side ith row element b_i by its corresponding group element g_0.

For any choice of row this gives us a mapping of the columns c^T_j into the real line (Mod 1).[19] The important point is that this

[17] This is often referred to as finite support.

[18] This approach was first described in (Gomory, 1969) section E and then developed in (Gomory & Johnson, 1972, Gomory & Johnson, 1972, Gomory & Johnson, 1973, Gomory & Johnson, 2003).

[19] Different row choices will give us different mappings and eventually different cutting planes.

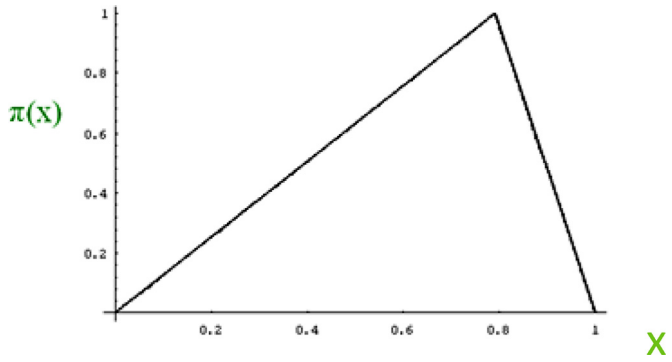

Fig. 12. Using a cutting plane.

mapping is addition preserving. So if the x_j^T are non-negative integers and are a solution to (6C) the group elements in RM1 corresponding to the columns c_j^T will form a path to g_0.

$$\sum_j x_j^N c_j^T \equiv b^T \quad (\text{Mod } 1) \qquad \text{maps into the path}$$

$$\sum_j x_j^N g\{j\} = g_0 \quad \text{in } RM1.$$

The non-negative integers x_j^T play the role of the t(g) in Eq. (7) in telling us how many times the group element is used in the path.

Now consider any cutting plane $(\pi\ (g), \pi_0)$ for the group RM1 with goal element g_0. Applying it to the path gives:

$$\sum_j \pi\,(g(j))\ x_j^T \geq \pi_0\ .$$

This is a cutting plane for the Corner Polyhedron.

So this a process that allows us to take our entire assortment of cutting planes for RM1 and turn them all into cutting planes for the Corner Polyhedron and therefore for the Integer programming problem from which they have arisen.

9.5. An arithmetic example of the process

Take the data from the *i*th row of a large Corner Polyhedron problem in the form of Eq. (6C):

$$4.72x_1^N - 2.93x_2^N \quad + 0.51x_3^N + 0.15x_4^N \cdots \equiv 2.79$$

We take the fractional parts of the row elements and of the right hand side:

0.72 0.07 0.51 0.15 ... 0.79

Fig. 12 shows here a cutting plane $(\pi\ (g), 1)$ for the Group RM1 with goal element 0.79.

In Fig. 12 we use the GMIC but it could just as well be any other group cutting plane for RM1 with goal element 0.79.

We enter our fractional values (they represent group elements) - on the x axis and read out their π values on the vertical axis.

We obtain 0.93, 0.09, 0.75, 0.21 ..... 1.00. The cutting plane inequality for the Corner Polyhedron produced using these values is:

$$0.93x_1^N + 0.09x_2^N \quad + 0.75x_3^N + 0.21x_4^N \ldots\ldots\ldots\ldots \geq 1.00$$

It is a valid cutting plane.

9.6. Other facets of the reals (Mod 1)

If we have other facets of RM1, we can use any of them in place of the GMIC in the cutting plane process of Section 9.5. Facets of RM1 are in fact very plentiful.

One very large collection of facets emerges from the Gomory–Johnson Theorem.[20]

Gomory–Johnson Theorem: If $\pi(x)$ has only two slopes and satisfies the condition $\pi(x)+\pi(1-x) = 1$ then it is a facet of RM1.

The x in the theorem refers to any group element located on the real interval [0, 1). The condition $\pi(x) + \pi(1-x) = 1$, which first appeared in the Structure Theorem, is also known as the symmetry or minimality condition.

In Fig. 13, we show some of the profusion of facets which emerge from this theorem. As in Fig. 11, the horizontal axis displays the group element in RM1 and the vertical axis displays the value of $\pi\,(x)$.

In these pictures lines of similar structure or color represent different facets $(\pi(x), 1)$.

In addition, if one of the facets in Fig. 12 was gradually moved to approach a neighboring one, all those new lines would also be facets.

In Fig. 13 the GMIC appears as the dotted line. But here the GMIC is not alone but part of a continuous family of related cutting planes. The GMIC, which originated in an ad hoc manner to deal with programming problems with both integer and continuous variables, and has a long track record as an effective practical cutting plane, now emerges as part of our theoretical structure.

In addition to the facets covered by the theorem there are many more. There are, for example, facets with three slopes, and there are ways of combining apparently different facets to make new ones (Gomory & Johnson, 2003).

Any of these facets can then be used by the process of Section 9.4 to generate cutting planes. The facet would appear in Fig. 12 in place of the GMIC.

So we are in a different world. We have a continuous profusion of facets and therefore the challenge of finding some way to select

[20] A proof of this theorem is also given in the supplementary material, Appendix B (Gomory & Johnson, 2003).

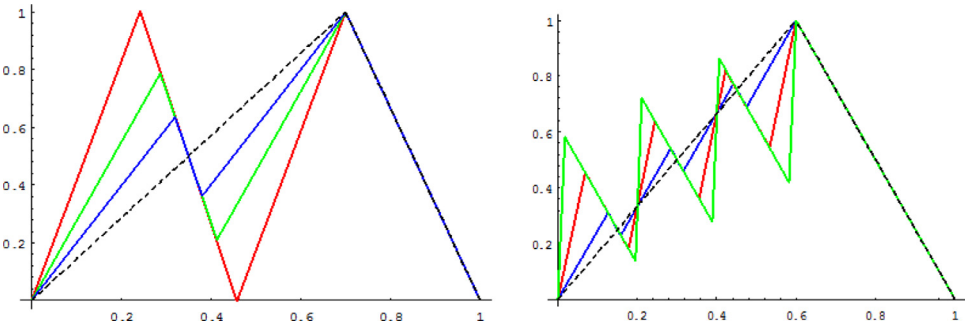

Fig. 13. Facets based on the Gomory–Johnson theorem.

among them. It would help to have an index that measures their power. This index, along with other closely related topics, is also addressed in (Gomory & Johnson, 2003).

10. Continuous variables and more dimensions

In this paper I have emphasized the early period and the motivation and problems that drove the creation and early evolution of Corner Polyhedra. As a result this article is already long, and two important directions, continuous variables and more dimensions, cannot be described here beyond a few remarks. However both areas will be discussed in Appendices that are available as supplementary material.

One result that is covered in that material is that the process of Section 9.4 for generating cutting planes extends simply and directly to take in continuous variables.[21]

However continuous variables can do more than merely repeat the results we have for pure integer problems, continuous variable problems have a life of their own. It is possible to start with continuous non-basic variables and develop cutting planes that then apply to the integer case as well. This process also sheds light on the special role of the GMIC.[22]

This direction turns out to be very helpful in developing cutting planes that are based not on a single row of an integer programming problem but on two or more rows. The best description of that area is in the many published papers of Gerard Cornuejols and Francois Margot.[23] My similar results in that area evolved directly from the approach and contents of this paper.[24]

11. Conclusion

Corner Polyhedra are a natural concept and we have been able to learn a surprising amount about them. What we already know suggests many directions for further exploration. We have seen that there are big facets and lots of tiny ones; how can we exploit this fundamental advantage? And the methods we use are limited, the simplex method is solving one set of linear equations after another, but we make no use at all of the classical methods for solving linear equations in integers. Also the Corner Polyhedra themselves provide a wonderful sequence of problems of increasing size whose structure is full of clues. Perhaps we should be solving maximization problems on Corner Polyhedra to see what it is about the structure that makes for computational difficulties and what makes things easy.

It has been a wonderful journey for me, from the paper mills, through periodicity, through the surprise of finding so much structure in the Structure Theorem and much more. When I think of the many possibilities for progress that we know about, not to mention those that have not yet revealed themselves, I feel confident that integer programming, as we know it today, is only at its earliest beginning.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2016.03.001.

References

Cornuejols, G., & Margot, F. (2009). On the Facets of mixed integer programs with two integer variables and two constraints. *Mathematical Programming, A, 120*, 429–456.
Gilmore, P. C., & Gomory, R. E. (1961). A linear programming approach to the cutting stock problem. *Operations Research, 9*(6)), 849–859 November-December.
Gilmore, P. C., & Gomory, R. E. (1963). A linear programming approach to the cutting stock problem - part II. *Operations Research, 11*(6), 863–888 November-December.
Gilmore, P. C., & Gomory, R. E. (1965). Multi-stage cutting stock problems of two and more dimensions. *Operations Research, 13*(1), 94–120 January-February.
Gilmore, P. C., & Gomory, R. E. (1966). The theory and computation of knapsack functions. *Operations Research, 14*(6), 1045–1074 November-December.
Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society, 64*(5), 275–278.
Gomory, R. E. (1967). *Faces of an integer polyhedron: 57* (pp. 16–18). Januaryalso published in Lectures in Applied Mathematics 2, Part I, Mathematics of the Decision Sciences, ed. by G.B. Dantzig and A.F.Veinott, Jr., American Mathematical Society, 1968, pp. 283-287.
Gomory, R. E. (1969). Some polyhedra related to combinatorial problems. *Journal of Linear Algebra and Its Applications, 2*(4), 451–558 October.
Gomory, R. E., & Abadie, J. (1970). Properties of a class of integer polyhedra. *Integer and non-linear programming* (pp. 353–365). North-Holland.
Gomory, R. E., & Johnson, E. L. (1972). Some continuous functions related to corner polyhedra - Part II. *Mathematical Programming, 3*(3), 359–389 North-Holland, December.
Gomory, R. E., & Johnson, E. L. (1972). Some continuous functions related to corner polyhedra. *Mathematical Programming, 3*(1), 23–85 North-Holland, August.
Gomory, R. E., & Johnson, E. L. (1973). The Group problems and subadditive functions. *Mathematical programming* (pp. 157–184). Academic Press.
Gomory, R. E., & Johnson, E. L. (2003). T-space and cutting planes. *Mathematical Programming, Ser. B 96* (pp. 341–375). Springer-Verlag.
Gomory, R. E., Johnson, E. L., & Evans, Lisa (2003). Corner polyhedra and their connections with cutting planes. *Mathematical Programming, Ser. B 96* (pp. 321–339). Springer-Verlag.
Wolfe, P., 0704 FORTRAN Mathematical Programming System, IBM Systems Reference Library, 1963, File No. 0704-0863RSM001. This is used with Michel Balinski, “Mathematical Programming System I”, ALL. IBM Systems Reference Library, 1963, File No. 0704-1092RSMIAS.

[21] Gomory(1970) is the source of these results. Appendix C of the supplementary material will be easier to read.
[22] Supplementary material, Appendix D.
[23] See for example (Cornuejols & Margot, 2009).
[24] Supplementary material Appendices D,E, and F