Living with *lrs*

David Avis

School of Computer Science
McGill University
3480 University, Montreal, Quebec, Canada H3A 2A7
avis@cs.mcgill.ca

ABSTRACT

This paper describes the development of *lrs*, an implementation of the reverse search method to the vertex enumeration/convex hull problem for convex polyhedra. We describe an important and difficult class of polyhedra, called configuration polytopes, that have application to determining the ground states of alloy phase diagrams. Experience gained while trying to solve these problems lead to a number of improvements to the original implementation.

1. Introduction

A classic result is that a convex polyhedron P can be represented in two ways. An *H-representation* is given by an $m \times d$ matrix $A = (a_{i,j})$ and m-vector $b = (b_i)$:

$$P = \{ y \in R^d \mid b + Ay \ge 0 \}. \tag{1}$$

We assume throughout that P is full dimensional and has at least one vertex, which implies that $m \ge d$. If A is minimal, that is no row can be deleted without changing P, then P has m *facets*, each defined by one of the inequalities in (1). A *vertex* $y \in R^d$ is a point of P that satisfies an affinely independent set of d inequalities as equations. An *extreme ray* $z \in R^d$ is a direction such that for some vertex y and any positive scalar t, $y + tz$ is in P and satisfies some set of $d-1$ affinely independent inequalities as equations. Note that an extreme ray is unique only up to a positive scalar, since if z is an extreme ray then so is tz for any positive scalar t. An equivalent *V-representation* of P is given by a minimal set of s *vertices* $y_1, \cdots, y_s$ and u *extreme rays* $z_1, \cdots, z_u$:

$$P = \{ y \in R^d \mid y = \sum_{i=1}^{s} \lambda_i y_i + \sum_{j=1}^{u} \mu_j z_j, \lambda_i \ge 0, \mu_j \ge 0, \sum_{i=1}^{s} \lambda_i = 1 \}. \tag{2}$$

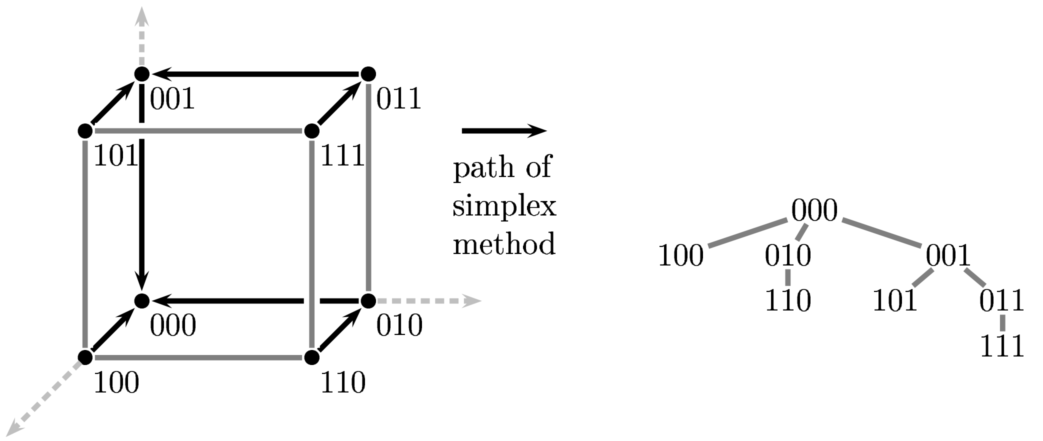
The *vertex enumeration problem* is to produce a V-representation from an H-representation, and the *facet enumeration problem* is to provide the reverse transformation. It is well known that these problems are essentially equivalent. Two classic methods exist for solving these problems: the double-description method (or insertion method), and pivot based methods. Further background and a description of these methods is contained in the books by Chvátal[9] and Ziegler[18].

The purpose of this paper is to describe experience gained in the development of *lrs*, *lexicographic reverse search* , a C program based on a pivoting method for vertex enumeration called *reverse search*. This method was first proposed by Fukuda and the author in 1992 [1]. Much of this experience was obtained in trying to use an early implementation of the code to solve a set of difficult problems arising from the so-called configuration polytopes, that are well known in materials science.

In this section we give a brief description of the method and some preliminary computational results on small problems. In the next section we describe the class of configurations polytopes, which appear to be computationally difficult. In Section 3 we discuss the lessons learned trying to enumerate their vertices. These lessons lead to various technical improvements to the basic algorithm, and have been incorporated into the program *lrs*. The final section gives some computational experience with this algorithm on various other well known polytopes. More complete technical descriptions of the methods and results described here appears in the papers [5] [6].

Briefly and informally, the reverse search algorithm works as follows. Suppose we have a system of m linear inequalities defining a d-dimensional polyhedron in R^d and a vertex of that polyhedron given by the indices of d inequalities whose bounding hyperplanes intersect at the vertex. These indices define a *cobasis* for the vertex. The complementary set of $m-d$ indices define a *basis*. For any given linear objective function, the simplex method generates a path between *adjacent* bases (or equivalently cobases) which are those differing in one index. The path is terminated when a basis of a vertex maximizing this objective function is found. The path is found by pivoting, which involves interchanging one of the hyperplanes defining the current cobasis with one in the basis. The path chosen from the initial given basis depends on the pivot rule used, which must be finite to avoid cycling. If we look at the set of all such paths from all bases of the polyhedron, we get a spanning forest of the graph of adjacent bases of the polyhedron. The root of each subtree of the forest is a basis of an optimum vertex. The reverse search algorithm starts at each root and traces out its subtree in depth first order by *reversing* the pivot rule.

The algorithm is particularly easy if each vertex lies on exactly d hyperplanes, and so has a unique basis. In this case the polyhedron is called *simple* or *non-degenerate*. The spanning forest has one component, which is a spanning tree of the skeleton of the polyhedron, and each vertex is produced once. An example of such a polyhedron is the cube, and Figure 1 shows a possible reverse search tree for it.



(a) The "simplex tree" induced by the objective $(-\sum x_i)$.

(b) The corresponding reverse search tree.

Figure 1: Reverse Search Tree for Vertex Enumeration of the Cube

The first implementation of the reverse search algorithm, *ve*01, was released by the author in 1992, and revised in 1994[3]. It solved the vertex enumeration problem for *polytopes* (bounded polyhedra), and did not report extreme rays in the case of unbounded polyhedra. A technical description of the latest implementation, *lrs*, is given in [6]. It handles unbounded polyhedra, reports extreme rays, does volume computation for facet enumeration problems, and estimates the output size and number of bases that will be computed. Both of the implementations use extended precision exact arithmetic.

In [3] some preliminary computational experience was given on a set of seven test problems. Table 1 shows the improvement in running time obtained between the original program *ve*01 and version 2.3 of *lrs* on these test problems. All of the problems *in*1,...,*in*7 are vertex enumeration problems, with #*F* input inequalities in *d* dimensions. In the table, #V and #R refer respectively to the number of vertices, and rays computed (*lrs* only).

Problem	facets	dim.	vertices	rays	*ve01*		*lrs*		*ve01/lrs*
	#F	d	#V	#R	bases	secs	bases	secs	time
in1	34	4	31		31	2.81	31	.06	46
in2	16	5	18		1247	7.48	76	.05	148
in3	19	6	8		10845	86.43	188	.12	720
in4	12	7	54		54	.43	54	.05	9
in5	14	9	89	33	97	1.14	94	.09	13
in6	23	10	332	302	3656	83.65	824	1.69	50
in7	20	10	1188		1188	208.98	1188	2.78	75

Table 1: Computational Results on Original Problem Set (*ve01 vs lrs***)**

Many of the improvements made were motivated by the solution of a set of considerably larger and more difficult problems. In the next section we describe these polytopes.

2. Configuration Polytopes

Alloy phase diagrams are of considerable interest in materials science and condensed matter physics. Physicists try to reproduce these diagrams and to understand their topology in terms of microscopic models for the energy and the entropy. An essential part of a first-principles phase diagram computation is the prediction of the ground states of the system. Lattice models with local interactions are used to predict which alloy structures can exist. One method, known as Kanamori's method is described in [15] and [11].

The idea of Kanamori's method is to derive a set linear constraints based on so-called correlation functions. The resulting polyhedron is called a *configuration polytope* and its vertices are the possible ground states of the required phase diagram. The correlation functions are derived from configurations on small clusters (due to the huge number of such constraints in general) and so in fact only an approximation of the exact configuration polytope is obtained. Therefore some vertices computed may violate constraints obtained from larger clusters, and cannot correspond to a physical state. These are known as *inconstructible* vertices. On the other hand, if a vertex in fact corresponds to some physical configuration, then it must be a vertex of the exact configuration polytope also. These are known as *constructable* vertices. Configuration polytopes pose a challenge to vertex enumeration programs, because even those based on very small clusters are highly degenerate.

In [8] Kanamori's method is applied to ternary alloys. Table 2 gives data on a number of configuration polytopes for this problem.

Problem	Upper Bound	Actual		Estimate			
		Vertices	Bases	Vertices	Bases	Probes	Nodes
729-9	1.15 e+10	4,862	477,421	2,032	160,907	3-20	9,941
31-20	587,860	18,553	169,272	11,596	120,743	2-20	7,519
41-16	8,544,096	29,108	287,806	69,827	204,209	2-20	5,917
71-61	5.47 e+09	3.15 e+06	5.76 e+07	2.32 e+06	5.18 e+07	2-5	25,765
90-86	3,067,078	323,188	1,621,760	272,195	1.53 e+06	2-5	30,534
288-281	9.16 e+12	?	?	2.91 e+11	2.90 e+12	1-1	1,322

Table 2: Configuration Polytopes

In the table, the first column gives the problem size $m - n$, where m is the number of linear inequalities and $d = n - 1$ is the dimension. Throughout the paper we refer to the problems by these numbers. The second column gives the maximum number of vertices of a polytope with this size, as given by McMullen's Upper Bound Theorem (see e.g.,[18]): a polyhedron P described by m inequalities in d dimensions has at most

$$f(m, d) = \binom{m - \lfloor \frac{d+1}{2} \rfloor}{m - d} + \binom{m - \lfloor \frac{d+2}{2} \rfloor}{m - d}.$$

vertices. The next two columns give the actual number of vertices and bases computed. The number of bases listed are the number of lexicographically positive bases of the polytope. The total number of bases is very much larger. The number of lexicographically positive bases is bounded by the Upper Bound Theorem, since is corresponds to the number of vertices of a perturbation of the original polytope to a simple polytope with the same number of inequalities. This issue is discussed in more detail below. The problem 288-281 has not yet been solved.

One difficulty of the vertex enumeration problem is the tremendous variability in the output size. The output may be as large as that given above in the Upper Bound Theorem, or as small as that given by the Lower Bound Theorem of Barnette (see e.g.,[18]): a non-empty simple polyhedron P described by m inequalities in d dimensions has at least

$$g(m, d) = m(d - 1) - (d + 1)(d - 2)$$

vertices. Non-simple polyhedra can have even fewer vertices, and general lower bounds were obtained by Deza and Fukuda[10] by inverting McMullen's condition. The bounds are tight and for most values of m and d there is obviously a very large gap between the lower and upper bounds. This motivated the search for a way to estimate the number of vertices and bases of a polyhedron without computing them all.

A feature of the reverse search method is that the reverse search tree can be probed at random. For any node in the tree it is possible to compute its degree, and then to generate a child uniformly at random. This allows the use of the Hall-Knuth estimator[13] for obtaining unbiased estimates of the number of nodes of a tree. The details of how this was done for *lrs* are given in [2].

The last four columns of the table refer to estimates obtained by *lrs* using this method. To limit the variance of the estimate, a complete search was done of the first few levels of the tree, then random probes were performed on all remaining subtrees. The notation $x - y$ in the second last column indicates that y independent random probes were made from each node of the tree at depth x. The final column gives the total number of tree nodes evaluated to produce the estimate. Although the estimates are unbiased, it can be shown that a low estimate is more likely than a high one, and this is born out by the results. Nevertheless, considering the huge gap between known upper and lower bounds for the number of vertices, the estimates are quite satisfactory. They are particularly useful for the purpose of deciding the tractability of the computation. The running time of the reverse search method is directly proportional to the number of bases given in the estimate. Since other enumeration methods store all of the vertices and/or bases computed, an estimate of their number is also usefull to determine the feasability of completing the computation. According to the estimates, it seems highly unlikely that problem 288-281 is tractable by any vertex enumeration method.

Except for problem 71-61, all results were obtained by running *lrs* on *mutt*, a DEC Alpha Server, as described in the Appendix. In the case of the polytope 71-61, the computation was performed by *prs*, a parallelized version of *lrs* described in [7]. This computation took 4.5 days on an NEC Senju computer with 64 processing elements. It is estimated that it would require 130 days to complete the computation on a single workstation. The output size is of the order of 1 gigabyte of data. This result and others in [7] demonstrate that the underlying reverse search algorithm is efficiently parallelizable.

3. Improvements to *lrs*

Several lessons were learned in working on the solution of the vertex enumeration problem for configuration polytopes. One was the necessity of providing restart capability, so that long runs that are terminated, voluntarily or involuntarily, can be restarted. Gerardo Garbulsky, who was using an early version of the program to solve 729-9, contacted the author about this after losing a month of computation after a power failure at MIT during the "storm of the century". In fact such a restart capability was quite straight forward to implement: only the cobasic indices of the current vertex of the reverse search tree need to be stored. The program is restarted by using the original input file and these indices to pivot the dictionary to the correct cobasis. The computation continues as before until all of the children of the root vertex have been explored.

Another important point demonstrated is the importance of rational arithmetic for combinatorial polytopes. For configuration polytopes, the rational coordinates of the vertices relate to the densities of various atoms in the underlying lattice, and it is therefore important that they be exact. A floating point solution to this problem was also obtained for 729-9, and in one case an automatic rounding function returned an incorrect solution. A second reason for using exact computation is the possibility of catastrophic error in the reverse search algorithm if a sign is calculated incorrectly. Since there is no redundancy in the underlying graph search, if some child of a given node in the tree is not discovered (due to an incorrect sign for example), then the entire subtree rooted at this node is not discovered either. Exact computation is more time consuming, and the next section gives some empirical results to show how much it costs.

A major improvement to *lrs* initiated by experience with configuration polytopes was the implementation of lexicographic perturbation. Without perturbation, a single vertex of the problem 729-9 has $\binom{120}{8}$ = 840261910995 bases. An earlier version of the program used numeric perturbation, and this allowed solution of the problem. However, with numeric perturbation there is no guarantee that a vertex is not lost after the perturbation. Furthermore, one has to deal numerically with the perturbed numbers, which tendto require more digits of precision than with the original data. Lexicographic pivoting is well known as a way of resolving degeneracy in linear programming (see e.g.,[14]). It is known to be equivalent to the epsilon perturbation method, however no perturbations are actually required and the original data is used at all times. The details of its implementation in *lrs* are given in [6]. An additional benefit in working with lexicographic pivoting is in the computation of volumes, as described in the next paragraph.

The problem of computing the volume of a polytope is a notoriously difficult problem, being #P-hard. If we consider the geometric interpretation of lexicographically positive bases for the facet enumeration problem, we see that they triangulate the facets of the polytope. It is therefore possible to determine the volume of the polytope from the determinants of the bases computed. Since *lrs* runs in linear space, this shows that the volume of a polytope can be computed in linear space if its vertices are known. It turns out that by using integer pivoting, as described in the next paragraph, the determinant of each basis is obtained at no extra cost during the pivot step. Therefore volume can be computed with almost no overhead during a facet enumeration with *lrs*. Volume computation was also combined with the estimation function, to provide unbiased estimates of the volume of a polytope given by a V-representation.

Profiles of the execution time of the earlier program *ve*01 indicated that approximately 80% of the running time was spent in greatest common divisor (gcd) computations. All calculation were performed in rational arithmetic, and after each arithmetic operation it was necessary to reduce the fraction to lowest terms. Gcd computations are typically proportional to the length of the integers involved, and hence are time consuming. An alternative to rational pivoting is integer pivoting, as described by Edmonds and Maurras[12] which is connected to Cramer's rule, see the appendix of Chvátal[9]. The essential idea is that the coefficients in the dictionary are the ratio of two integer determinants, the denominator being the determinant of the current basis. It is therefore only necessary to store the numerators of the dictionary coefficients and the common denominator (which can be used in volume computation as remarked above.) As documented in [5] integer pivoting improves the running time by between 1.3 and 18 times, depending on the problem. Using integer pivoting the number of digits computed can get and remain quite large, increasing the time for basic arithmetic operations. On the other hand, no gcd computations are required. Interestingly the greatest speedups were found to occur with the problems requiring the longest integers. For example, cyclic polytopes required 164 decimal digits in the computation, which ran 18 times faster with integer pivoting than with rational pivoting. Even with integer pivoting, the cost of the the computation using extended precision arithmetic is dominated by the pivot step in the algorithm. Roughly 85-90% of the time is used in this step.

Another lesson learned in developing *lrs* involved its generalization to the related convex hull problem. From a theoretical standpoint, convex hull and vertex enumeration problems are essentially equivalent. They are precisely equivalent for polytopes that contain the origin. From a theoretical point of view, one may assume the input is given in this form. This seems a reasonable assumption, since the input data for a polytope can always be translated so that this is true, at the cost of solving a linear program if no feasible point is known. However, in practice this translation is inconvenient, since the output results must also be translated. Furthermore, the translation depends on a knowing an interior point of the polytope. Even if such a point is known, the translation may involve a large increase in the size of the integers required in the computation. In the case of unbounded polyhedra further complications arise.

Another method to solve the convex hull problem is to lift the polyhedron to one higher dimension, where it becomes a pointed cone. This standard homogenization technique is described for example in [18]. pp. 30-31. The "vertex enumeration" problem for cones becomes a ray enumeration problem, which is trivially equivalent to the facet enumeration for the dual cone. In order to use this method, the program had to be extended to handle unbounded polyhedra. Initially it was thought that this lifting method would be unsatisfactory, since a cone has a single completely degenerate vertex at the origin. Surprisingly the lifted problems ran faster, even when the unlifted polytope was non-degenerate. As described in [5] empirically the lifted polyhedron had roughly half the number of bases as the unlifted one. As a simple example, consider lifting the square in the plane to a cone with four faces in space. After perturbation, the cone has only two bases, whereas the square, even though non-degenerate, has four. Mattheis [17] reports similar findings for his lifting technique, which differs from the homogenization used in *lrs*, since it results in a polyhedron (rather than a cone) in one higher dimension, that has additional vertices. The number of additional vertices created by this lifting technique may be large, see Klee [16].

A final improvement made in developing *lrs* was in the optional use of additional memory. As mentioned earlier, some configuration polytopes require up to a gigabyte of space just to store the output. A feature of reverse search is that this output does not have to be stored in memory, or even saved off-line. Only the dictionary for the current basis is needed. We have also observed that most of the computation time is used in pivoting. In pure reverse search, the tree is explored by pivoting to new bases, and backtracking is performed by pivoting back to the parent node. In this way a stack is not required. An obvious decrease in running time can therefore be obtained by storing all dictionaries in the path from the root to the current basis. The amount of space required depends on the length of this path, which in theory can be long. In fact the length of the path is just the number of pivots required by the simplex method to reach the optimum basis starting at the current basis. The Klee-Minty and similar examples (see e.g.[9]) show that in the worse case this path can contain all of the vertices in the polyhedron. However in practice, such long path lengths are not encountered. Using a circular queue, *lrs* stores a predetermined number of parent dictionaries on the path back to the root. If the parent dictionary is stored, it can be reloaded without pivoting saving up to half of the total number of pivots.

The improvements described above were mainly motivated by the attempt to solve the special class of configuration polytopes. In the next section we see how they improve the running time for a large variety of different kinds of polyhedra.

4. Other Polyhedra

In this section we show the result of the improvements made to *lrs* on various other polyhedra. A complete description of the results is contained in [5]. Some empirical results are given for a set of four classes of polyhedra that have been well studied in the literature: cut polyhedra, cyclic polyhedra, Kuhn-Quandt polyhedra, metric polyhedra. The problems were chosen to include both simple and highly degenerate polytopes, and to require a wide range of digits of precision in the calculations. Each problem is given a name *name* m_n, where m and n denote the size of the input array for the problem. For vertex enumeration, m is the number of input inequalities, and for facet enumeration it is the number of vertices. n is the dimension of the polytope plus one. Data for the seven problems chosen is given in Table 3, and the input files are available on-line as described in the appendix. The first three problems and the last one are facet enumeration problems, and the others are vertex enumeration problems. For each problem the output to be computed is highlighted in bold face. The volume is given for the facet enumeration problems. Although the volume is computed exactly, a floating point approximation is given here due to the large size of the integers.

Several of these problems are essentially unsolvable by pivot methods without perturbation. The unperturbed *cut*32_16 has 44450496 bases, and the unperturbed *metric*40_11 has 115972624 bases. It is not known how many bases *metric*80_16 has.

Problem	input rep.	dim. d	facets #F	vertices #V	volume	*lrs* bases	*lrs* secs
cut16_11	V	10	**56**	16	2.3116...	124	0.13
cut32_16	V	15	**368**	32	1.3457...	46751	169.67
cyclic25_13	V	12	**35700**	25	4.84e+75	18564	909.23
kq20_11	H	10	20	**1188**	-	1188	3.84
metric40_11	H	10	40	**32**	-	9184	17.78
metric80_16	H	15	80	**544**	-	6450702	37532.68

Table 3: Other Polyhedra

This confirms that theoretical results obtained in[4] are observed on polyhedra arising in practice.

For the problems in Table 3, lifting improves the running time by a factor roughly between 2 and 5, and integer pivoting improves the running time by a factor roughly between 1.3 and 20. It was possible to solve the *cut* and *metric* problems using a fixed precision version of the program prepared by Ambros Marzetta. This enables us to get some estimate of the overhead used by the general extended precision integer processing. These were solved by a 64-bit version of the program, which ran between 4.2 and 6.3 times faster than the general extended precision program.

5. Acknowledgements

The author would like to thank many people for help on this project. Gerardo Garbulsky introduced him to the fascinating configuration polytopes. Komei Fukuda worked with the author on the solution of the problems, solving them independently with his *cdd* code based on the double description method. Ambros Marzetta supplied the author with a fixed precision version of *lrs*, and programmed a parallel version of *lrs*. David Bremner and Jerry Quinn helped program some of the improvements to the *lrs* code. An anonymous referee pointed out the reference to Mattheis' lifting method.

6. Appendix

The program *lrs*, and the input files for the polyhedra described in this paper can be obtained from the author's home page http://www.cs.mcgill.ca/~avis. The configuration polytope input files were provided by Gerardo Garbulsky. Except where noted, the results in the tables were obtained by *mutt*, a DEC AlphaServer 1000 4/23 at McGill.

References

1. D. Avis and K. Fukuda, "A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra," *Discrete and Computational Geometry*, vol. 8, pp. 295-313, 1992.
2. D. Avis and L. Devroye, "Estimating the Number of Vertices of a Polyhedron," in *Snapshots of Computational and Discrete Geometry*, ed. D. Avis and P. Bose, vol. 3, pp. 179-190, School of Computer Science, McGill University, 1994. ftp://mutt.cs.mcgill.ca/pub/doc/avis/AD94a.ps.gz
3. D. Avis, "A C Implementation of the Reverse Search Vertex Enumeration Algorithm," in *RIMS Kokyuroku 872*, ed. H. Imai, Kyoto University, May 1994. ftp://mutt.cs.mcgill.ca /pub/doc/avis/Av94a.ps.gz
4. D. Avis, D. Bremner, and R. Seidel, "How Good are Convex Hull Algorithms?," *Computational Geometry: Theory and Applications*, vol. 7, pp. 265-301, 1997.

5. D. Avis, "Computational Experience with the Reverse Search Vertex Enumeration Algorithm," *Optimization Methods and Software*, vol. 10, pp. 107-124, 1998.
6. D. Avis, *lrs*: A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm, May 1998. ftp://mutt.cs.mcgill.ca /pub/doc/avis/Av98a.ps.gz
7. A. Brüngger, A. Marzetta, K. Fukuda, and J. Nievergelt, *The Parallel Search Bench ZRAM and its Applications*, 1997. ftp://ftp.ifor.math.ethz.ch/pub/fukuda/reports
8. G. Ceder, G.D. Garbulsky, D. Avis, and K. Fukuda, "Ground States of a Ternary Lattice Model with Nearest and Next-Nearest Neighbor Interactions," *Physical Review B*, vol. 49, pp. 1-7, 1994.
9. V. Chvátal, *Linear Programming,* W.H. Freeman, 1983.
10. A. Deza and K. Fukuda, "McMullen's Conditions and some Lower Bounds for General Convex Polytopes," *Geometriae Dedicata*, vol. 52, pp. 165-173, 1994.
11. F. Ducastelle, in *Order and Phase Stability in Alloys*, North-Holland, 1991.
12. J. Edmonds and J.-F. Maurras, "Note sur les Q-matrices d'Edmonds," *Recherche Opérationelle (RAIRO)*, vol. 31, pp. 203-209, 1997.
13. M. Hall and D.E. Knuth, "Combinatorial Analysis and Computers," *Am. Math. Monthly*, vol. 72, pp. 21-28, 1965.
14. J. Ignizio and T. Cavalier, *Linear Programming,* Prentice Hall, 1994.
15. M. Kaburagi and J. Kanamori, "A Method of Determining the Ground State of the Extended Range Classical Lattice Gas Model," *Progress in Theoretical Physics*, vol. 54, pp. 30-44, 1975.
16. V. Klee, "Polytope Pairs and Their Relationship to Linear Programming," *Acta Math.*, vol. CXXXIII, pp. 1-25, 1974.
17. T.H. Matheiss, "An Algorithm for Determining Irrelevant Constraints and All Vertices in Systems of Linear Inequlaties," *Operations Research*, vol. 21, pp. 247-260, 1973.
18. G. Ziegler, *Lectures on Polytopes,* Springer, 1994, revised 1998. Graduate Texts in Mathematics.