# ESTIMATING THE NUMBER OF VERTICES OF A POLYHEDRON

*David Avis**

*Luc Devroye*

School of Computer Science

McGill University

3480 University

Montréal, Québec, Canada

H3A 2A7

*Dedicated to Godfried Toussaint, a convex hull pioneer*

*ABSTRACT*

Given a polyhedron $P$ by a list of inequalities we develop unbiased estimates of the number of vertices and bases of $P$. The estimates are based on applying tree estimation methods to the reverse search technique. The time to generate an unbiased estimate is essentially bounded by the time taken to solve a linear program on $P$ with the simplex method. Computational experience is reported. The method can be applied to estimate the output size of other enumeration problems solvable by reverse search.

## 1. Introduction

Let $A$ be an $m$ by $n$ matrix with $m \geq n$ and let $b$ an $n$-vector. The *convex polyhedron* $P$ is defined by

$$P = \{ x \in R^n | Ax \leq b \}. \tag{1.1}$$

For definitions not given here and further information, the reader is referred to Chapter 2.3, written by Bayer and Lee, of the handbook [10]. We assume throughout that $A$ has full column rank,

* Research supported by N.S.E.R.C. grant number A3013, F.C.A.R. grant number EQ1678.

that $P$ is non-empty and that there are no redundant inequalities. A bounded polyhedron is called a *polytope*. In this note we will deal with polytopes rather than possibly unbounded polyhedra. A point $\bar{x} \in P$ is a *vertex* of $P$ if there is some $n$ by $n$ submatrix $B$ of $A$ such that $\bar{x}$ is the unique solution of $Bx = b$. The matrix $B$ is called a *basis* for $\bar{x}$. The vertex is *degenerate* if more than $n$ of the inequalities in (1.1) are satisfied as equations by $\bar{x}$. In other words it can be represented by more than one basis. A polytope with no degenerate vertices is called *simple*.

The well known Upper Bound Theorem of McMullen (see [10] ) states that the polyhedron $P$ has at most

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

vertices. On the other hand, the Lower Bound Theorem of Barnette (see [10] ) states that a non-empty simple polytope $P$ has at least

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

vertices. Non-simple polyhedra can have even fewer vertices, and general lower bounds were obtained by Deza and Fukuda [7] by inverting McMullen's condition. The bounds are tight and for most values of $m$ and $n$ there is a very large gap between the lower and upper bounds.

An important computational problem is to generate all of the vertices of $P$. There are essentially two main approaches to this problem, both with there origins in the 1950s. The double description (or Motzkin) method [13] involves building the polytope sequentially by adding the defining inequalities one at a time. Recent algorithms of this type have been developed by Seidel, see [8], and Chazelle [5]. This method seems very powerful for degenerate polytopes but has the disadvantage that it can require a lot of memory. A practical implementation has been developed by Fukuda [9]. The second method of vertex generation involves pivoting around the edge skeleton of the polytope (for general references, see [6]. ) Specifically, a graph is defined on the bases of $P$. Two bases are adjacent they differ in exactly one row. A search of this graph generates all bases and hence all vertices. An efficient practical method using this approach is the reverse search method of the Avis and Fukuda [2]. Their approach is to compute directly a spanning tree of the basis graph. The method has time complexity depending on the number of bases of $P$ and so is very slow for highly degenerate polytopes. The advantage of the reverse search method is that it requires only the space required to store the polytope $P$, and does not require the bases to be stored. A practical implementation of this method has been developed by Avis [3].

Let $V$ and $N$ denote respectively the number of vertices and bases of $P$. At present there is no known method of computing the vertices of $P$ in time polynomial in $m, n$ and $V$. The pivoting methods are, however, polynomial in $m, n$ and $N$. For example, the reverse search method generates all bases in time $O(n(m-n)\min(m-n, n)N)$ and space $O(mn)$. In order to determine the feasibility of a vertex generation problem it is therefore necessary to have good estimates of $N$ and $V$.

Due to the availability of the practical implementations of vertex generation algorithms various large problems have now been solved. For instance, in [4] all vertices of a non-simple polytope with $m = 729$ and $n = 8$ were found. Using the Upper Bound theorem we can see that this polytope has at least 17 and at most 11,479,694,949 vertices. This is of little use in determining the feasibility of generating the vertices of the given polyhedron. In fact the polytope has 4862 vertices and is highly degenerate. This result was obtained by both the double description [9] and reverse search [3] methods, requiring many days of computation on a SPARC work station. The time taken by reverse search is proportional to the number of bases of the polytope. For a degenerate polytope, the number of bases is reduced by perturbation of the defining half-spaces. For this polytope, a perturbation yielded a polytope with 477,471 bases which was generated on a SPARC work station in about three weeks, using exact integer arithmetic.

The above example shows the need to be able to estimate the number of vertices and bases of a polytope, both before and after perturbation, and the need to estimate the running time of an algorithm. In this paper we give a method for estimating the number of vertices and bases of a polytope, and for estimating the running time of the reverse search method. We describe a class of randomized algorithms that give estimates whose expected value is the correct answer.

Although the variance is high, methods are suggested for reducing it. The basic idea is to estimate the size of the reverse search tree using tree estimation techniques. These are described in the next section, which also includes a discussion of how to reduce the variance. In Section 3 we discuss the application of these results to the vertex estimation problem for convex polytopes and give some experimental results on estimates for configuration polytopes.

## 2. Tree Estimation

Perhaps the simplest estimate of the size of a tree is the Hall-Knuth (HK) estimator [11] which is analyzed in detail in Knuth [12]. This estimate is formed as follows. For a vertex $v$ in a tree $T$, let $n(v)$ be the number of children of $v$. A random path

$$v_0,\ v_1,\ \ldots,\ v_k$$

is generated such that:

(i) $v_0$ is the root and $v_k$ is a leaf of $T$, and

(ii) for $i = 1, \ldots, k$, $v_i$ is a child of $v_{i-1}$ chosen randomly with probability $\dfrac{1}{n(v_{i-1})}$.

An estimate $\mathbf{N}$ of the number of vertices of $T$ is given by

$$\mathbf{N} = 1 + \sum_{i=0}^{k-1} n(v_0)n(v_1)\ldots n(v_i).$$

This estimate can be understood as the size of a tree where each vertex at depth $i$ from the root has the same number of children as $v_i$. Hall and Knuth showed that the expected value $\bar{\mathbf{N}}$ of $\mathbf{N}$ is the number of nodes in $T$, showing that the estimate is *unbiased*. The *cost* of an estimator is defined to be the number of tree nodes evaluated in obtaining the estimate. The cost of the HK estimator is $k + 1$, the number of nodes on the random path. In the worst case it is one more than the height of the tree.

In order to use this estimator, it is necessary to find a random path in a tree. In general, given a graph $G$ it is not possible to find a random path in the depth-first or breadth-first spanning tree of $G$ without actually computing the tree. The HK estimator has been used primarily for a relatively limited number of well structured backtracking examples. Reverse search, however, allows a random path to be generated. This is because, given any node of the tree, it is possible to generate all children of this node without any other information. A direct application of the HK estimator to the basis graph of a polytope using reverse search to generate a random path gives an estimate of the number of bases. This is discussed further in the next section.

The major problem with the HK estimator is that the estimate has a high variance. Some methods to reduce the variance are described in [12]. In the next three subsections, we discuss ways to reduce the variance.

**Multiple probes from the root.**

The simplest way to reduce the variance is simply to make a number $q$ of independent estimates $\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_q$ from the root. The estimate $\mathbf{M}(q)$ for $T$ is

$$\mathbf{M}(q) = \frac{1}{q} \sum_{i=1}^{q} \mathbf{N}_i$$

which is clearly unbiased. If the probes are independent the variance of $\mathbf{M}(q)$ is given by

$$Var\ \mathbf{M}(q) = \frac{Var\ \mathbf{N}}{q}.$$

The cost of computing $\mathbf{M}(q)$ is $q$ times the cost of computing a single estimate $\mathbf{N}$.

**Single probe with look-ahead.**

The HK estimator tends to underestimate the size of the tree with fairly high probability. Consider a large binary tree with one of the two nodes attached to the root a leaf. With probability 1/2 the HK estimate is 3. In order to avoid these low estimates it is possible to "look-ahead" in the tree and avoid paths that lead to "small" subtrees. This can be done as follows.

Let $d$ be a non-negative look-ahead parameter. The path is generated from the root by choosing the next vertex at random from among those children which have descendants at depth $d$. The selection is biased towards the child with the most descendants at depth $d$. If no such children exist the path is terminated, otherwise the procedure is repeated from the selected child.

This procedure combines features of methods studied by Knuth [12], and the proof that the estimate is unbiased can be derived along similar lines.

For a tree $T$ and any vertex $v$ in the tree, let $n_d(v)$ be the number of descendants at exactly depth $d$ from $v$. If $T$ has depth at least $d$, a random path

$$v_0, v_1, \ldots, v_k$$

is generated such that

(i) $v_i$ is a child of $v_{i-1}$ and is chosen with probability $\dfrac{n_{d-1}(v_i)}{n_d(v_{i-1})}$ from the children of $v_{i-1}$, where $i = 1, \ldots, k$, and

(ii) $v_k$ is the first vertex in the path with $n_d(v_k) = 0$.

Let

$$m(v) = 1 + \sum_{\substack{w \text{ is a child of } v \\ n_{d-1}(w) = 0}} |\ \textit{subtree rooted at } w\ |.$$

Note that in particular $m(v_k)$ is the size of the subtree rooted at $v_k$. The estimate $\mathbf{L}(d)$ of the size of the tree is given by

$$\mathbf{L}(d) = m(v_0) + \sum_{i=1}^{k} \frac{n_d(v_0) n_d(v_1) \cdots n_d(v_{i-1})}{n_{d-1}(v_1) n_{d-1}(v_2) \cdots n_{d-1}(v_i)}\ m(v_i).$$

For $d = 1$ this is the HK bound. The cost of computing $\mathbf{L}(d)$ increases greatly with $d$ in general, and depends on the average degree of nodes in $T$.

**Single probes from depth $s$.**

Using the notation of the previous section, $T$ has $n_s(v_0)$ nodes at depth $s$. As an alternative to making $t = n_s(v_0)$ probes from the root $v_0$, we may make one probe from each node at depth $s$. Let $\mathbf{N}(v)$ be the HK estimate obtained for the subtree rooted at node $v$. Then the depth $s$ estimate $\mathbf{D}(s)$ is

$$\mathbf{D}(s) = \sum_{i=0}^{s-1} n_i(v_0) + \sum_{v \text{ is a node at depth } s} \mathbf{N}(v).$$

For $s = 0$ this is the HK estimate. With $t = n(v_0)$ the cost of computing an estimate $\mathbf{M}(t)$ is comparable to that of computing $\mathbf{D}(1)$ since the same number of random probes are computed. However the latter estimate has lower variance, as made precise in the following theorem.

**Theorem 2.1**

Set $t = n(v_0)$ and let $\mathbf{M}(t)$ and $\mathbf{D}(1)$ be estimates of the size of a tree $T$.

(i) $Var\ \mathbf{M}(t) \geq Var\ \mathbf{D}(1)$ with equality holding if and only if all subtrees of $T$ at depth 1 have the same cardinality.

(ii) $E\ cost\ \mathbf{M}(t) \geq E\ cost\ \mathbf{D}(1)$ with equality holding if and only if the root has degree 1.

**Proof:** Label the children of the root $u_1, u_2, \ldots, u_t$. Now

$$\mathbf{D}(1) = 1 + \sum_{i=1}^{t} \mathbf{N}(u_i),$$

where $\mathbf{N}(u_i)$ is the HK estimate of the size of the subtree rooted at $u_i$. Since the random variables $\mathbf{N}(u_i)$ are independent,

$$Var\ \mathbf{D}(1) = Var\ \sum_{i=1}^{t} \mathbf{N}(u_i) = \sum_{i=1}^{t} E[\mathbf{N}(u_i) - \bar{\mathbf{N}}(u_i)]^2.$$

Now let $\mathbf{N}$ be a HK estimate from the root $v_0$. Conditioned on the event that $u_i$ is on the random path, $\mathbf{N}$ has the same distribution as $1 + t\mathbf{N}(u_i)$. From this we observe that the expected value $\bar{\mathbf{N}}$ of $\mathbf{N}$ is given by

$$\bar{\mathbf{N}} = 1 + \sum_{i=1}^{t} \bar{\mathbf{N}}(u_i).$$

The variance of $\mathbf{N}$ is given by

$$E[\mathbf{N} - \bar{\mathbf{N}}]^2 = \sum_{i=1}^{t} E[(\mathbf{N} - \bar{\mathbf{N}})^2 \mid u_i \textit{ is on path }] \, \mathbf{Pr}(u_i \textit{ is on path })$$

$$= \frac{1}{t} \sum_{i=1}^{t} E[1 + t\mathbf{N}(u_i) - \bar{\mathbf{N}}]^2$$

$$= t \sum_{i=1}^{t} E[\mathbf{N}(u_i) - \frac{1}{t} \sum_{j=1}^{t} \bar{\mathbf{N}}(u_j)]^2$$

$$\geq t \sum_{i=1}^{t} E[\mathbf{N}(u_i) - \bar{\mathbf{N}}(u_i)]^2$$

$$\geq t \; Var \; \mathbf{D}(1).$$

Now the result follows since

$$Var \; \mathbf{M}(t) = \frac{Var \; \mathbf{N}}{t} \geq Var \; \mathbf{D}(1).$$

For the costs, we observe that

$$E \; cost \; \mathbf{M}(t) = t \; cost \; \mathbf{N} = t(1 + \frac{1}{t} \sum_{i=1}^{t} E \; cost \; \mathbf{N}(u_i))$$

$$\geq 1 + \sum_{i=1}^{t} E \; cost \; \mathbf{N}(u_i) = E \; cost \; \mathbf{D}(1).$$

□

**Tail estimates.**

A tail estimate is useful in determining the feasibility of a vertex generation problem. If $\mathbf{X}$ is an unbiased estimate of the size of a tree $T$, then by Markhov's inequality, for every positive constant $c$,

$$\mathbf{Pr}( \; |T| > \frac{\mathbf{X}}{c} \; ) \; > \; 1 - \frac{1}{c} \, . \tag{1}$$

This tail estimate can be used with any of the preceding estimators as a way of discarding hopeless problems. However, it does not give assurances that when $\mathbf{X}$ is small the problem is probably tractable. For this one requires bounds on the other tail of $\mathbf{X}$. At present we have no such bounds for any of the above estimators.

**Red-Blue trees**

For the vertex generation application, a modified estimation problem is also of interest. Assume that $T$ has nodes coloured either red or blue. It is required to estimate the number of red vertices. Again, let

$$v_0, v_1, \ldots, v_k$$

be a random path in the tree. Then the HK estimate for the number of red nodes, $\mathbf{R}$ , is given by

$$\mathbf{R} = \mathbf{1}_{\{v_0 \ red\}} + \sum_{i=0}^{k-1} \mathbf{1}_{\{v_{i+1} \ red\}} n(v_0) \cdots n(v_i),$$

where $\mathbf{1}_{\{\}}$ is the indicator function. It is easy to show that this is an unbiased estimate of the number of red nodes. Similar modifications apply to the three variance reduction methods described above.

## 3. Vertex Estimation

To apply the methods of the previous section, one must be able to compute all children of any given tree vertex. The reverse search method, *rs*, allows such a search and so all of the methods of the previous section apply. For the vertex generation problem, *rs* generates a spanning tree of all bases of an input polytope $P$. If $P$ is simple, each basis corresponds to a unique vertex. Hence the estimates of the previous section immediately give estimates for the number of vertices of $P$. Note that the worst case cost of obtaining a HK estimate is the maximum path length + 1 in the reverse search tree. This path length is the worst case number of pivots required by the simplex method using Bland's rule to go from a feasible basis to the optimum basis. The Klee-Minty examples are known to generate an exponential number of pivots under Bland's rule [1], but this behaviour is rather pathological in practice.

In the case where $P$ is non-simple, the same vertex may be represented by many bases. Since a basis is a subset of $n$ inequalities from the $m$ inequalities defining $P$, we can represent it by an $n$ element subset of integers from $1, \ldots, m$. For a degenerate vertex, the $lex-min$ basis is the lexicographically smallest subset of indices that define the vertex. In the reverse search tree for a non-simple polytope $P$, we can distinguish two types of nodes. Those that correspond to lex-min bases(red) and those that do not(blue). The number of vertices in $P$ is exactly the number of red nodes in $T$. Using the techniques of the last section, we can in principle estimate the number of vertices of non-simple polytopes. In practice, for highly degenerate polytopes these estimates will have enormous variance.

For non-simple polytopes *rs* is usually applied not to the original polytope but to a perturbed polytope. Using perturbation a simple polytope is obtained with fewer bases than the original polytope. If the perturbation is valid, the bases of the perturbed polytope are a subset of the original bases which contain all of the vertices of the original polytope. Applying the estimator to the perturbed polytope, we can obtain an estimate the running time of *rs*. This is due to the fact that *rs* runs in time linear in the number of bases. For any given problem, one can run *rs* and quickly determine the running time per basis. From this an unbiased estimate of the total running time is readily obtained. Then it is possible to decide whether or not it is likely that *rs* will be able to complete the calculation.

In Table 3.1 we give some experimental results on configuration polytopes that were supplied by Gerardo Garbulsky. Each problem is numbered $m-n$ giving the number $m$ of inequalities and the dimension $n$. They are all highly degenerate. Column 2 gives the upper bound on the number of vertices given by the Upper Bound Theorem. Where known, the actual number of vertices are given in column 3. A perturbation was applied to each polytope. Column 5 gives the number of vertices in the perturbed polytope for the three cases solved by *rs*. The final 3 columns give estimates of the number of vertices of the perturbed polytope. The estimates were obtained by the depth $s$ probe method, by *mutt* a DEC Alpha work station. The entry $s-x$ in the "Probes" column indicates the estimate was obtained by averaging $x$ depth $s$ estimates. The "Nodes" column gives the total number of tree nodes explored to obtain the estimate given in column 5. The estimate in column 5 should be compared with the actual value (where known) given in column 4. Incidentally, using the current version of *rs*, *mutt* estimates that she would need about 16 weeks to solve 71-60, 19 days to solve 90-85, and 377 thousand years to solve 288-281, but these estimates are probably optimistic. As an application of the tail inequality (1) of the previous section, with probability at least .999 it would take at least 377 years to solve 288-281 with this program. For those looking for harder problems, Geraldo Garbulsky has problems 554-545 and 842-743 available on request (email: gdg@lanai.mit.edu).

| Problem | Upper Bound | # Vertices | # Vertices (Perturbed) | Estimate # Vertices | Probes | Nodes |
|---|---|---|---|---|---|---|
| 729-8 | 1.147 e+10 | 4,862 | 477,421 | 2.41 e+05 | 3-20 | 11,580 |
| 31-19 | 587,860 | 11,850 | 168,739 | 3.47 e+05 | 2-20 | 12,856 |
| 41-15 | 8,544,096 | 29,108 | 318,602 | 1.80 e+05 | 2-20 | 8,382 |
| 71-60 | 5.471 e+09 | >115,980 | ? | 1.97 e+07 | 2-5 | 29,073 |
| 90-85 | 3,067,078 | ? | ? | 1.63 e+06 | 2-5 | 42,782 |
| 288-280 | 9.164 e+12 | ? | ? | 2.38 e+12 | 1-3 | 3,974 |

**Table 3.1: Configuration Polytopes**

**References**

1. D. Avis and V. Chvátal, "Notes on Bland's Pivoting Rule," *Math. Programming Study*, vol. 8, pp. 24-34, 1978.

2. 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.

3. D. Avis, "A C Implementation of the Reverse Search Vertex Enumeration Algorithm," *Research Report, RIMS, Kyoto University*, June 1993. (Revised version of: Technical Report SOCS-92.12, McGill University, School of Computer Science)

4. 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.

5. B. Chazelle, "An Optimal Convex Hull Algorithm and New Results on Cuttings," *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, pp. 29-38, 1991.

6. V. Chvátal, *Linear Programming,* W.H. Freeman, 1983.

7. A. Deza and K. Fukuda, "McMullen's Conditions and some Lower Bounds for General Convex Polytopes," *Geometriae Dedicata*, 1994(to appear).

8. H. Edelsbrunner, *Algorithms in Combinatorial Geometry,* Springer-Verlag, 1987.

9. K. Fukuda, *DD.p: A Pascal Implementation of the Double Description Method,* Version 2.0, Graduate School of Systems Management, Tsukuba University, 1993.

10. P.M. Gruber and J.M. Wills, *Handbook of Convex Geometry,* North-Holland, 1993.

11. M. Hall and D.E. Knuth, ‘‘Combinatorial Analysis and Computers,’’ *Am. Math. Monthly*, vol. 72, pp. 21-28, 1965.

12. D.E. Knuth, ‘‘Estimating the Efficiency of Backtrack Programs,’’ *Mathematics of Computation*, vol. 29, pp. 121-136, 1975.

13. T.S. Motzkin, H. Raiffa, G.L. Thompson, and R. M. Thrall, ‘‘The Double Description Method,’’ *Annals of Math. Studies 8*, pp. 51-73, Princeton University Press, 1953.