Appl. Math. Lett. Vol. 4, No. 5, pp. 39–42, 1991
Printed in Great Britain.

0893-9659/91 $3.00 + 0.00

A Basis Enumeration Algorithm for Linear Systems with Geometric Applications

DAVID AVIS[1] AND KOMEI FUKUDA[2]

[1]School of Computer Science, McGill University
[2]Graduate School of Systems Management, The University of Tsukuba

(Received February 1991)

Abstract. We present a new pivot-based algorithm which can be used with minor modification for the enumeration of all bases, or all feasible bases, of a linear system. The algorithm has the following properties: no additional storage is required beyond the input data; the output list produced is free of duplicates; the running time is output sensitive for non-degenerate inputs; the algorithm is easy to efficiently parallelize. It can be used for various geometric enumeration problems including finding all facets of the convex hull of a set of points and the enumeration of all vertices of a convex polyhedron or hyperplane arrangement, and can be extended to oriented matroids.

Let A be a $m \times n$ matrix, with columns indexed by the set $E = \{1, 2, ..., n\}$. Fix distinct indices f and g of E. Consider the system of equations:

$$A\,x \;=\; 0,\; x_g \;=\; 1. \tag{1}$$

For any $J \subseteq E$, x_J denotes the subvector of x indexed by J, and A_J denotes the submatrix of A consisting of columns indexed by J. A *basis* B for (1) is a subset of E of cardinality m containing f but not g, for which A_B is nonsingular. We will only be concerned with systems (1) that have at least one basis, and will assume this for the rest of the paper. Given any basis B, we can transform (1) into the *dictionary*:

$$x_B \;=\; -A_B^{-1}A_N\,x_N \;=\; \bar{A}\,x_N,$$

where $N = E - B$ is the *co-basis*, and $\bar{A}$ denotes $-A_B^{-1}A_N$. $\bar{A}$ is called the *coefficient matrix* of the dictionary, with rows indexed by B and columns indexed by N, so that $\bar{A} = (\bar{a}_{ij} \;:\; i \in B,\; j \in N)$. Note that the co-basis always contains the index g.

A variable x_i is *primal feasible* if $i \in B - f$ and $\bar{a}_{ig} \geq 0$. A variable x_j is *dual feasible* if $j \in N - g$ and $\bar{a}_{fj} \leq 0$. A dictionary is *primal feasible* if x_i is primal feasible for all $i \in B - f$ and *dual feasible* if x_j is dual feasible for all $j \in N - g$. A dictionary is *optimal* if it is both primal and dual feasible. A *basic solution* to (1) is obtained from a dictionary by setting $x_{N-g} = 0$, $x = 1$. If any basic variable has value zero, we call the basic solution and corresponding dictionary *degenerate*. In this note we outline an algorithm for enumerating all distinct basic solutions of the system (1) without repetition, using only the space required to store the input. Full details and proofs are given in the report [1]. The algorithm is initiated with an optimal dictionary. A variant of the algorithm enumerates all primal feasible dictionaries reporting the corresponding basic feasible solutions without repetition.

Several geometric problems can be transformed to dictionary enumeration problems. A (convex) polyhedron P is the solution set to a system of n_0 inequalities in d non-negative

The work of the first author was performed while visiting the laboratory of Professor Masakazu Kojima of Tokyo Institute of Technology, supported by the JSPS/NSERC bilateral exchange program.

variables: $P = \{y \in R^d \mid A'y \leq b,\ y \geq 0\}$, where A' is an $n_0 \times d$ matrix and b is a n_0-vector. A *vertex* of the polyhedron is a vector $y \in P$ that satisfies a linearly independent set of d of the inequalities as equations. The *vertex enumeration problem* for P is to enumerate all of its vertices. In fact to find even a single vertex of P is computationally equivalent to linear programming. As we wish to separate this from the enumeration problem, we will assume we are given an initial vertex. An optimal dictionary for P can be constructed with $m = n_0 + 1$ and $n = n_0 + d + 2$. It can be shown that each *primal feasible* dictionary has a basic solution which gives a vertex y of P. By employing a standard duality between points and hyperplanes, we may transform the facet enumeration problem for the convex hull of a set of points into a vertex enumeration problem for a convex polyhedron. Similarly the algorithm can be used to enumerate the vertices of a Voronoi diagram and cells of a Delaunay triangulation. The variant of the algorithm that lists all dictionaries may be used to enumerate the vertices of an arrangement of hyperplanes. For surveys of other approaches to these problems, the reader is referred to [2–4].

Suppose we are given a system of equations of the form (1), for some $m \times n$ matrix A. The *linear programming problem* (LP) for (1) is to maximize x_f over (1) subject to the additional constraint that each variable except x_f and x_g is non-negative. Each optimal dictionary is a solution to LP. To begin with, we will assume that there is a unique optimal dictionary. A *pivot* $(r,\ s)$ on a basis B, and corresponding dictionary $x_B = \bar{A}x_B$, is an interchange of some $r \in B - f$ with some index $s \in N - g$ giving a new basis B'. The new coefficient matrix $A' = (a'_{ij})$ is given by

$$a'_{rs} = -\frac{1}{\bar{a}_{rs}},\quad a'_{is} = \frac{\bar{a}_{is}}{\bar{a}_{rs}},\quad a'_{rj} = \frac{\bar{a}_{rj}}{\bar{a}_{rs}},\quad a'_{ij} = \bar{a}_{ij} - \frac{\bar{a}_{is}\bar{a}_{rj}}{\bar{a}_{rs}},\quad (\, i \in B - r,\ j \in N - s\,).$$

The pivot is *primal feasible* (respectively, *dual feasible*) if both of the dictionaries corresponding to B and B' are primal (respectively, dual) feasible. The simplex method is a method of solving LP by beginning with an initial dictionary and pivoting until an optimal dictionary is found. We consider two rules for choosing a pivot. The first rule, known as Bland's rule starts with primal feasible basis B.

BLAND'S RULE.

(1) Let s be the smallest index such that x_s is dual infeasible, that is, $\bar{a}_{fs} > 0$.
(2) Set $\lambda = \min\{-\frac{\bar{a}_{ig}}{\bar{a}_{is}} : i \in B - f,\ \bar{a}_{is} < 0\}$. Let r be the smallest index obtaining this minimum.

The pivot $(r,\ s)$ maintains the primal feasibility of the dictionary. The second rule, known as the criss-cross rule, starts with any basis.

CRISS-CROSS RULE.

(1) Let $i \neq f,\ g$ be the smallest index such that x_i is (primal or dual) infeasible.
(2) If $i \in B$, let $r = i$ and let s be the minimum index such that $\bar{a}_{rs} > 0$, otherwise let $s = i$ and let r be the minimum index such that $\bar{a}_{rs} < 0$.

The criss-cross pivot $(r,\ s)$ interchanges x_r and x_s, and may not preserve either primal or dual feasibility. In both cases, if step (1) does not apply then the dictionary is optimal.

The validity of these rules is given by the following proposition. Part (a) is proved in [5] and part (b) in [6] for linear programs, and in [7,8] in the more general setting of oriented matroids. A simple proof of part (b) also appears in [9].

PROPOSITION 1. *Let (1) be a system that admits an optimal dictionary and let B be any basis.*

(a) If B is primal feasible, then successive application of Bland's rule leads to an optimal dictionary, and each basis generated is primal feasible.

(b) Successive application of the criss-cross rule starting with basis B leads to an optimal dictionary.

We now outline a dictionary enumeration algorithm for systems (1) that admit a unique optimal dictionary. Consider a graph where vertices are dictionaries and two vertices are adjacent if the corresponding two dictionaries differ in only one basic variable. Then part (b) of the proposition tells us that there is a unique path consisting of criss-cross pivots from any dictionary to the optimal dictionary. The set of all such paths gives us a spanning tree in this graph. Consider a non-optimal dictionary D with basis B. Let (r, s), $r \in B - f$, $s \in N - g$, be the pivot obtained by applying the criss-cross rule to D giving a dictionary D'. We call (s, r) a *reverse criss-cross pivot* for D'. Suppose we start at the optimal dictionary and explore reverse criss-cross pivots in lexicographic order. This corresponds to a depth first search of the spanning tree defined above. When moving down the tree, each dictionary is encountered exactly once. In order to backtrack, the algorithm simply performs a criss-cross pivot on the current dictionary to return to the parent in the tree. From here the algorithm continues by considering the next candidate reverse pivot in lexicographical order. A similar analysis applies to part (a) of the proposition. We form a similar graph, except that vertices are just the primal feasible dictionaries. We define a *reverse Bland pivot* in the analogous way. A depth first search of this graph provides all primal feasible dictionaries.

The efficiency of the procedure depends greatly on efficiently checking candidate reverse pivots. To test whether a coefficient matrix $\bar{A}$ arises from a coefficient matrix A' by a criss-cross (resp., Bland) pivot interchanging row r with column s of A', it is only necessary to examine rows f, r and columns g, s of A'. These can be computed from $\bar{A}$ in $O(m + n)$ time, and checked to see if (r, s) is a criss-cross (resp., Bland) pivot. Further savings are possible, as certain potential reverse pivots can be eliminated without any pivoting.

PROPOSITION 2. *If (s, r), $s \in B - f$ and $r \in N - g$, is a valid reverse criss-cross pivot for a dictionary $x_B = \bar{A}x_N$, then either*

(a) $\bar{a}_{sg} > 0$, $\bar{a}_{sr} > 0$, $\bar{a}_{sj} \geq 0$ *for* $j \in N - g$, $j < s$, *or*
(b) $\bar{a}_{fr} < 0$, $\bar{a}_{sr} < 0$, $\bar{a}_{ir} \leq 0$ *for* $i \in B - f$, $i < r$.

PROPOSITION 3. *Let $x_B = \bar{A}x_N$ be a dictionary, let $r \in N - g$ and set $\lambda = min\{-\frac{\bar{a}_{ig}}{\bar{a}_{ir}} : i \in B - f, \bar{a}_{ir} < 0\}$. If (s, r), $s \in B - f$, is a valid reverse Bland's rule pivot then s must be an index that obtains this minimum.*

We use a lexicographic procedure to ensure that each basic solution is output exactly once, based on the following proposition.

PROPOSITION 4. *Let B be a basis for a degenerate dictionary $x_B = \bar{A}x_N$. B is not lexicographically minimum for the corresponding basic solution if and only if there exists $r \in B - f$ and $s \in N - g$ such that $r > s$, $\bar{a}_{rg} = 0$ and $\bar{a}_{rs} \neq 0$.*

The procedure outlined above will only generate all (feasible) dictionaries if the system (1) has a unique optimal dictionary. In case there are many optimal dictionaries, we modify the procedure to use the dual form of Bland's rule to construct all optimal dictionaries to the original problem. Details are given in the full paper [1].

We now discuss the complexity of the dictionary enumeration algorithm. Let $f(A)$ denote the number of dictionaries that can represent (1). For each dictionary, we may evaluate $(m - 1)(n - m - 2)$ candidates for reverse pivots, each candidate requiring $O(m + n)$ time. Therefore the overall time-complexity of the enumeration algorithm is

$$O\Big((m+n)\ m\ (n-m)\ f(A)\Big) = O\left((m+n)\ m\ n \binom{n-2}{m-1}\right).$$

Apart from a few indices, no additional space is required other than that required to represent the input. Let $g(A)$ denote the number of primal feasible dictionaries representing (1). The above analysis and (2) hold, with $g(A)$ replacing $f(A)$. In the non-degenerate case we can do better. Recalling Proposition 3, we see that we only need to consider one candidate reverse pivot per column of the dictionary: if there are two or more indices realizing the minimum then a pivot would give a degenerate dictionary. For each column, the candidate

basic variable can be found by computing the minimum ratio λ in $O(m)$ time. To check if a candidate is in fact a reverse pivot, we need to construct the objective row of the dictionary after the pivot, taking $O(n - m)$ time. Therefore since there are $n - m - 2$ candidate columns, all reverse Bland pivots from the given dictionary can be found in $O((n - m)n)$ time, in the non-degenerate case. This gives an overall complexity of $O((n - m)ng(A))$ for the non-degenerate case.

Consider now the enumeration of the vertices of a polyhedron given by a list of n_0 inequalities in d variables. Since we assume the polyhedron has at least one vertex, $n_0 \geq d$. We have $m = n_0 + 1$ and $n = n_0 + d + 2$. The time-complexity of enumerating all of the vertices is $O(n_0^2 d\ g(A))$. Again the complexity is output sensitive for non-degenerate polyhedra, for which $g(A)$ is just the number of vertices. If the polyhedron is simple (i.e., all dictionaries are non-degenerate) then we get an improved complexity bound. The algorithm produces vertices at a cost of $O(n_0\ d)$ per vertex with no repetitions and no additional space. As far as we know, this is the first solution to this problem that does not require exponential space. These complexities apply to the convex hull problem, where n_0 is the number of input points. In the non-degenerate case where no more than d points lie on any facet (i.e., the facets are simplicial), we can enumerate the v facets in time $O(n_0 dv)$ and space $O(n_0 d)$. For the vertex enumeration of an arrangement of n_0 hyperplanes in R^d, the complexity is $O(n_0^2 df(A))$ time and $O(n_0 d)$ space.

REFERENCES

1. D. Avis and K. Fukuda, A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra, *Tech. Rept. B-237*, Tokyo Institute of Technology, Dept. of Information Science (November 1990).
2. V. Chvátal, *Linear Programming*, W.H. Freeman, (1983).
3. M.E. Dyer, The complexity of vertex enumeration methods, *Math. Oper. Res.* **8**, 381–402 (1983).
4. H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, (1987).
5. R.G. Bland, A combinatorial abstraction of linear programming, *J. Combin. Theory B* **23**, 33–57 (1977).
6. T. Terlaky, A convergent criss-cross method, *Math. Oper. und Stat. ser. Optimization* **16**, 683–690 (1985).
7. T. Terlaky, A finite criss-cross method for oriented matroids, *J. Combin. Theory B* **42**, 319–327 (1987).
8. Z. Wang, A conformal elimination free algorithm for oriented matroid programming, *Chinese Annals of Mathematics* **8** (B,1) (1987).
9. K. Fukuda and T. Matsui, On the finiteness of the criss-cross method, *European J. O. R.* (to appear).

[1]School of Computer Science, McGill University, 3480 University, Montreal, Quebec, Canada H3A 2A7
[2]Graduate School of Systems Management, The University of Tsukuba, Otsuka, Bunkyo-ku, Tokyo 112, Japan