Symmetrized-lexicographic order and the method of points

Harold R. Parks *

Department of Mathematics, Oregon State University, Corvallis, OR 97331-4605, USA

Received 27 December 2005
Available online 30 January 2006
Submitted by Steven G. Krantz

Abstract

A variant of lexicographic order called symmetrized-lexicographic order is defined. The symmetrized-lexicographic order finds its application in the goal programming procedure called the method of points. The symmetrized-lexicographic order is shown to be representable using linear algebra and, thus, the method of points can be implemented as a linear programming problem.

Keywords: Symmetrized-lexicographic order; Method of points; Goal programming; Linear programming

1. Introduction

Symmetrized-lexicographic order is a linear ordering on equivalence classes of words: Given a word, all words that can be formed by permuting the letters are equivalent and the equivalence classes are lexicographically ordered based on the permutation in which the letters are in non-decreasing order. Experience has shown that the idea can be most easily grasped by looking at a very simple example: Consider the words “cat,” “sat,” and “tar.” In the next array, each row contains all the permutations of the letters in one of the words, so each row is an equivalence class. Each row has been put in lexicographic order so the left-most entry will have its letters in non-

* Fax: +1 541 737 0517.
E-mail address: parks@math.orst.edu.
decreasing order. Finally, the rows have been ordered so that first column itself in lexicographic order, thus putting the equivalence classes in symmetrized-lexicographic order.

\[
\text{act atc cat cta tac tca}
\]

\[
\text{art atr rat rta tar tra}
\]

\[
\text{ast ats sat sta tas tsa}
\]

In this paper, we consider the symmetrized-lexicographic order on an initial segment of the natural numbers. In the main mathematical result, Corollary 4, we prove that the symmetrized-lexicographic order can be realized using linear algebra and the usual linear ordering of the natural numbers.

As an application, we describe the goal programming method called the method of points in which the symmetrized-lexicographic order is used. The method of points was created to help with the task of making teaching assignments in the Oregon State University mathematics department. The symmetrized-lexicographic order was the natural choice in that context. Corollary 4 allows us to reduce the application of the method of points, which is potentially a large combinatorial problem, to a linear programming problem.

2. Symmetrized-lexicographic order

We establish some notation.

Notation 1.

1. The natural numbers will mean \( \mathbb{N} = \{1, 2, \ldots \} \).
2. Let \( \mathcal{N}^k \) denote the set of non-decreasing \( k \)-tuples of natural numbers. That is, \((a_1, a_2, \ldots, a_k) \in \mathcal{N}^k \) if and only if \((a_1, a_2, \ldots, a_k) \in \mathbb{N}^k \) and \( a_1 \leq a_2 \leq \cdots \leq a_k \).
3. Let \( \prec \) denote the lexicographic ordering on \( \mathbb{N}^k \) or \( \mathcal{N}^k \). That is, \((a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k) \)

if and only if there is \( 1 \leq i \leq k \) so that \( a_j = b_j \) holds for \( j < i \) and \( a_i < b_i \).
4. For \( n_1, n_2 \in \mathbb{N} \), we let \( \mathbb{N}^k[n_1, n_2] \) denote the subset of \( \mathbb{N}^k \) in which all the entries are between \( n_1 \) and \( n_2 \), that is,

\[
\mathbb{N}^k[n_1, n_2] = \mathbb{N}^k \cap \{(a_1, a_2, \ldots, a_k) : n_1 \leq a_i \leq n_2 \text{ for } i = 1, 2, \ldots, k\}.
\]

Similarly, let \( \mathcal{N}^k[n_1, n_2] \) denote the subset of \( \mathcal{N}^k \) in which all the entries are between \( n_1 \) and \( n_2 \), so

\[
\mathcal{N}^k[n_1, n_2] = \mathcal{N}^k \cap \mathbb{N}^k[n_1, n_2].
\]
Definition 2.

1. Let $I: \mathbb{N}^k \to \mathbb{N}^k$ be the map that puts the entries of $(a_1, a_2, \ldots, a_k)$ into non-decreasing order, that is, for $(a_1, a_2, \ldots, a_k) \in \mathbb{N}^k$, $I(a_1, a_2, \ldots, a_k)$ is defined by requiring
   
   (a) $I(a_1, a_2, \ldots, a_k) \in \mathbb{N}^k$ and
   (b) $I(a_1, a_2, \ldots, a_k)$ is a permutation of $(a_1, a_2, \ldots, a_k)$.

2. The symmetrized-lexicographic order on $\mathbb{N}^k$, denoted by $\prec_{\text{sym}}$, is defined by requiring
   
   $(a_1, a_2, \ldots, a_k) \prec_{\text{sym}} (b_1, b_2, \ldots, b_k)$ if and only if $I(a_1, a_2, \ldots, a_k) \prec I(b_1, b_2, \ldots, b_k)$.

The definition of symmetrized-lexicographic order is more easily expressed if we separate the idea of permuting the entries from the notion of ordering, thus we have introduced the space $\mathbb{N}^k$ and the function $I$.

Theorem 3. Fix a number $K \in \mathbb{N}$. For each natural number $n$, there exists

$$ f : \{1, 2, \ldots, n\} \to \mathbb{N} $$

with the property that, for $k = 1, 2, \ldots, K$ and

$$(a_1, a_2, \ldots, a_k), (b_1, b_2, \ldots, b_k) \in \mathbb{N}^k[1, n],$$

it holds that

$$(a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k) \text{ if and only if } \sum_{i=1}^{k} f(a_i) < \sum_{i=1}^{k} f(b_i). \quad (2.1)$$

Proof. We will prove the result by induction on $n$. Note that if $n = 1$, then the result is trivial, since neither side of (2.1) can be true. For completeness, we define $f(1) = 1$.

Now suppose that the result holds for a particular natural number $n$. We will show that the result also holds for $n + 1$.

Let $\tilde{f} : \{1, 2, \ldots, n\} \to \mathbb{N}$ be the function whose existence is guaranteed by the induction hypothesis, and define $g : \{1, 2, \ldots, n + 1\} \to \mathbb{N}$ by setting

$$ g(i) = \begin{cases} 1 & \text{if } i = 1, \\ \tilde{f}(i - 1) & \text{if } 2 \leq i \leq n + 1. \end{cases} $$

Then $g$ has the property that, for $k = 1, 2, \ldots, K$ and

$$(a_1, a_2, \ldots, a_k), (b_1, b_2, \ldots, b_k) \in \mathbb{N}^k[2, n + 1],$$

it holds that

$$(a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k) \text{ if and only if } \sum_{i=1}^{k} g(a_i) < \sum_{i=1}^{k} g(b_i). \quad (2.2)$$
Set
\[ d_1 = g(1) - g(2), \]
\[ d_2 = g(1) + g(n + 1) - 2 \cdot g(2), \]
\[ \vdots \]
\[ d_k = g(1) + (k - 1) \cdot g(n + 1) - k \cdot g(2), \]
\[ \vdots \]
\[ d_K = g(1) + (K - 1) \cdot g(n + 1) - K \cdot g(2) \]
and set
\[ D = 1 + \max\{0, d_1, d_2, \ldots, d_K\}. \]

Define \( f : \{1, 2, \ldots, n + 1\} \to \mathbb{N} \) by setting
\[ f(i) = \begin{cases} 
1 & \text{if } i = 1, \\
D + g(i) & \text{if } 2 \leq i \leq n + 1.
\end{cases} \]

We claim that \( f \) satisfies (2.1).

By (2.2) and because \( f(i) = D + g(i) \) holds for \( 2 \leq i \leq n + 1 \), \( f \) has the property that, for \( k = 1, 2, \ldots, K \) and
\[ (a_1, a_2, \ldots, a_k), (b_1, b_2, \ldots, b_k) \in \mathcal{N}^k[2, n + 1], \]
it holds that
\[ (a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k) \quad \text{if and only if} \quad \sum_{i=1}^{k} f(a_i) < \sum_{i=1}^{k} f(b_i). \tag{2.3} \]

Suppose \( k \in \{1, 2, \ldots, K\} \) and
\[ (a_1, a_2, \ldots, a_k), (b_1, b_2, \ldots, b_k) \in \mathcal{N}^k[1, n + 1]. \]

There are four cases to consider:

(1) \( a_1 = b_1 = 1 \),
(2) \( a_1 = 1 \) and \( 2 \leq b_1 \),
(3) \( 2 \leq a_1 \) and \( b_1 = 1 \), and
(4) \( 2 \leq a_1, b_1 \).

First, we will look separately at the argument that applies when \( k = 1 \).

Case (1): In case (1), neither \( a_1 \prec (b_1) \) nor \( f(a) < f(b) \) holds.

Case (2): In case (2), we have \( (a_1) \prec (b_1) \) by definition. We have \( f(a_1) = 1 \) and \( f(b_1) = D + g(b_1 - 1) \geq 1 + f(b_1 - 1) \geq 2 \), so \( f(a_1) < f(b_1) \) holds.

Case (3): Arguing similarly to case (2), we see that both \( (b_1) \prec (a_1) \) and \( f(b_1) < f(a_1) \) hold.

Case (4): In case (4), we appeal to (2.3) which holds on \( \{2, 3, \ldots, n + 1\} \).

Note that, when \( k = 1, \) (2.3) is equivalent to the assertion that \( f \) is an increasing function.

We now may suppose that \( 2 \leq k \leq K \) holds.

Case (1): In case (1),
\[ (a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k) \]
is equivalent to
\[(a_2, a_3, \ldots, a_k) \prec (b_2, b_3, \ldots, b_k).\]

By (2.3), which holds on \([2, 3, \ldots, n + 1],\)
\[(a_2, a_3, \ldots, a_k) \prec (b_2, b_3, \ldots, b_k)\]
is equivalent to
\[\sum_{i=2}^{k} f(a_i) < \sum_{i=2}^{k} f(b_i).\]

Since \(f(a_1) = f(b_1) = f(1) = 1,\) it follows that
\[\sum_{i=1}^{k} f(a_i) < \sum_{i=1}^{k} f(b_i)\]
is equivalent to
\[(a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k).\]

Case (2): In case (2),
\[(a_1, a_2, \ldots, a_k) \prec (b_1, b_2, \ldots, b_k)\]
holds. Because \(f\) is an increasing function, we have
\[\sum_{i=1}^{k} f(a_i) \leq f(1) + (k - 1) f(n + 1) = g(1) + (k - 1) D + (k - 1) g(n + 1)\]
and
\[\sum_{i=1}^{k} f(b_i) \geq kf(2) = kD + kg(2).\]

So it holds that
\[\sum_{i=1}^{k} f(b_i) - \sum_{i=1}^{k} f(a_i) \geq D + kg(2) - g(1) + (k - 1) g(n + 1)\]
\[\geq 1 + d_k + kg(2) - g(1) - (k - 1) g(n + 1) = 1.\]

Case (3): Arguing similarly to case (2), we see that \((b_1, b_2, \ldots, b_k) \prec (a_1, a_2, \ldots, a_k)\) and
\[\sum_{i=1}^{k} f(b_i) < \sum_{i=1}^{k} f(a_i)\]
hold.

Case (4): In case (4), we can simply apply (2.3). \(\square\)

The summations on the right-hand side of (2.1) are, of course, commutative, so the order of the entries in \((a_1, a_2, \ldots, a_k)\) and \((b_1, b_2, \ldots, b_k)\) does not matter. The usefulness of restricting to \(\mathcal{N}^k\) was in organizing the proof of Theorem 3.
Corollary 4. Fix a number $K \in \mathbb{N}$. For each natural number $n$, there exists

$$f : \{1, 2, \ldots, n\} \to \mathbb{N}$$

with the property that, for $k = 1, 2, \ldots, K$ and

$$(a_1, a_2, \ldots, a_k), (b_1, b_2, \ldots, b_k) \in \mathbb{N}^k [1, n],$$

it holds that

$$(a_1, a_2, \ldots, a_k) \prec_{\text{sym}} (b_1, b_2, \ldots, b_k) \text{ if and only if } \sum_{i=1}^{k} f(a_i) < \sum_{i=1}^{k} f(b_i). \quad (2.4)$$

Remark 5. The example that started this paper can be covered by Theorem 3 with $K = 3$ and $n = 5$, since the words are three letters long and involve only the five letters a, c, r, s, and t. The proof of the theorem gives the function $f(1) = 1$, $f(2) = 19$, $f(3) = 25$, $f(4) = 27$, $f(5) = 28$ and, hence, the correspondence

$$a \mapsto 1, \quad c \mapsto 19, \quad r \mapsto 25, \quad s \mapsto 27, \quad t \mapsto 28.$$

3. The method of points

One of the more daunting tasks one might undertake is the job of assigning a set of $n$ indivisible items to $n$ individuals, henceforth called players. The method of points has been developed to facilitate this task so as to minimize the unfairness of the distribution. The presumption is that the various players have disparate opinions regarding the desirability of the items to be distributed, and thus it might be possible for all players to be satisfied with the assignment as a whole, even if no individual player receives his or her most desired item.

Though the situation is different and the result of applying it is typically not fair (in the technical sense), the procedure described below, and called the method of points, took its inspiration from the adjusted-winner procedure of Brams and Taylor [1]. Also, we remark that the method of points might be considered to be a variation on Chebyshev goal programming [2]. The method was created to deal with the practical problem of making teaching assignments.

Procedure: The same number of points is allotted to each player, and each player is to use those points to indicate the non-negative integer value that player places on each of the indivisible items.

The $i$th player’s assignment of points will be represented by the row vector

$$v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,n}).$$

We will call $v_i$ the $i$th preference vector. The preference matrix is

$$V = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n,1} & v_{n,2} & \cdots & v_{n,n} \end{pmatrix}.$$
Full preference information is given by the augmented matrix
\[
\begin{pmatrix}
v_{1,1} & v_{1,2} & \ldots & v_{1,n} & 1 \\
v_{2,1} & v_{2,2} & \ldots & v_{2,n} & 2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
v_{n,1} & v_{n,2} & \ldots & v_{n,n} & n
\end{pmatrix}.
\]

An assignment of the items to the players is represented by a permutation \(\sigma : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}\), where the \(i\)th item goes to the \(\sigma(i)\)th player. The payoff to the \(\sigma(i)\)th player is then \(v_{\sigma(i),i}\). We can represent the assignment using the augmented matrix
\[
\begin{pmatrix}
v_{\sigma(1),1} & v_{\sigma(1),2} & \ldots & v_{\sigma(1),n} & \sigma(1) \\
v_{\sigma(2),1} & v_{\sigma(2),2} & \ldots & v_{\sigma(2),n} & \sigma(2) \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
v_{\sigma(n),1} & v_{\sigma(n),2} & \ldots & v_{\sigma(n),n} & \sigma(n)
\end{pmatrix}.
\]

The payoff vector is the vector with the numbers on the main diagonal as its entries:
\[
(v_{\sigma(1),1}, v_{\sigma(2),2}, \ldots, v_{\sigma(n),n}).
\]

The \(i\)th player’s payoff is the \(i\)th element along the main diagonal which is also the \(i\)th entry in the payoff vector.

The location of the minimum diagonal element tells us which player received his or her least desired assignment. Since the value of that minimum indicates the level of satisfaction of the recipient with that least desired assignment, it is taken as axiomatic that we should maximize the value of the minimum diagonal element. Arguing inductively, we see that among assignments that maximize the minimum diagonal element, we should select one that maximizes the second smallest diagonal element, and so on. Thus we have the following:

**Premise.** To obtain the most equitable assignment, we should maximize the payoff vector in the symmetrized-lexicographic order.

Since there are \(n!\) permutations, finding the assignment that maximizes the payoff vector by brute force can only be done when \(n\) is small. The motivation for the main result, Corollary 4, was to show that we can reduce the optimal assignment problem for the method of points to a linear programming problem; thus, interior-point methods can be applied.

**Theorem 6.** Maximizing the payoff vector in the symmetrized-lexicographic order can be expressed as a linear programming problem.

**Proof.** The payoff vector for a particular permutation is the diagonal of the matrix obtained by left-multiplying \(V\) by the corresponding permutation matrix. By applying Corollary 4, we see that we can replace each \(v_{i,j}\) by the \(f(v_{i,j})\) and maximize the sum of the diagonal elements. \(\square\)

**References**
