A General Theory of Social Choice

Abstract

In this paper we give a more elaborate development of our algorithm for social choice which generalizes Condorcet's method and satisfies Arrow's five criteria and two axioms as proven in the companion paper, *A Social Choice Algorithm (1997)*. An example is worked out for the case of m, the number of alternatives, equals 5. The steps of the algorithm itself are precisely explicated. A proof is presented that the algorithm provides a social choice solution for all values of m when individual and social choices are expressed in terms of the R (preference or indifference) operator, and for all values of n, the number of voters. The demonstration of a Social Welfare Function (SWF) and the proof that it provides solutions in all cases proves that social choice is indeed possible.

As was shown in the companion paper, *A Social Choice Algorithm*, the key to proving social choice possible is the admission of ties as possible solutions. In the cases for which there are tie solutions, an additional criterion can be used to winnow the solution set. We introduce the concept of "digital utility" to choose among the various ties the one or ones that have the best "goodness of fit" with the voters' preferences.

Introduction

In our previous paper, *A Social Choice Algorithm*, we presented an algorithm which generates solutions (social choices) which satisfy Arrow's criteria and axioms. A social choice may consist of a single element or of a set of tie elements and is based solely on binary, pairwise comparisons. If a candidate dies, the solution generated by considering the individual voting lists with the dead candidate's name blotted out and recomputing the solution is entirely consistent with the solution generated by taking the original solution and blotting out the dead candidate's name. These solutions are also entirely consistent with the Condorcet solution when a singular solution exists and represent a generalization of the Condorcet solution when the solution consists of a set of ties. The algorithm considered here, therefore, is a generalization of the Condorcet method.

In *A Social Choice Algorithm* we showed that solutions of the kind generated by our algorithm satisfy Arrow's criteria. In this paper we give a more elaborate and precise definition of the algorithm and then go on to prove that it provides solutions for all values of m and n where m is the number of alternatives and n is the number of voters. Therefore, social choice is possible if tie solutions are considered.

In this paper we will use the standard R operator where xRy means x "is prefered or indifferent to" y. R with a subscript refers to an individual voter and without the subscript refers to society. Therefore, xR_iy represents an individual choice, and xRy represents a social choice. A social welfare function (SWF) maps the domain of individual choices of the n voters and m alternatives into the range of social choices.

For the binary case, Arrow [1951] considers ties possible. His Axiom I states: "For all x and y, either xRy or yRx." He goes on to say "...Axiom I does not exclude the possibility of both xRy and yRx." Therefore, there are three possible solutions in the binary case: xRy, yRx and {xRy, yRx}. Arrow hints that the tie solution would imply xIy where I is the indifference operator. We would like to maintain the distinction between a social tie and a social indifference and, therefore, introduce the tie operator T. xTy means that there is a tie between xRy and yRx. However, the following analysis would proceed in an identical fashion if T were interchanged with I.

Majority rule is used to determine the results in the binary case. Let N(x,y) be the number who vote xRy and let N(y,x) be the number who vote yRx. The decision rule is the following: The social choice is xRy if N(x,y) > N(y,x) and yRx if N(y,x) > N(x,y). If N(x,y) = N(y,x), the social choice is xTy.

The SWF considered here is based on the observation that for the voting paradox in which there are three voters with the following votes, aR_1bR_1c, bR_2cR_2a and cR_3aR_3b, none of the following, aRbRc, bRcRa, cRaRb, are satisfactory as a social choice since, for example, if aRbRc is chosen, two out of the three voters have cR_ia which contradicts the social choice which implies aRc. However, if the tie set {aRbRc, bRcRa, cRaRb} is considered as the social choice (which would seem obvious in this case), this contradiction need not occur. We know that this tie set must reduce to aRb, bRc and cRa for the binary cases. Blotting out a b from each member of the set, {aRbRc, bRcRa, cRaRb}, leaves us with the set, {aRc, cRa, cRa} which we know reduces to cRa. Two out of the three members of the set are cRa which suggests that two out of the three voters prefer c to a. Therefore, we combine the two cRa and drop the element aRc. Generalizing this notion for reducing the solution for m=3 to the one for m=2, we take the tie set which is the solution at stage m, blot out a particular letter from each element and then combine elements according to the following rule. Add together all similar elements. If for some element the total is greater than any of the other elements, this element is the solution for stage m-1. If there is a set of elements all of which have the same total, and this total is greater than the total for any other element, then this set becomes the solution at stage m-1. A more complete discussion is given in *A Social Choice Algorithm.*.

Please note that Arrow did not consider this kind of "rule" for going from stage m to stage m-1 because he did not consider the possibility that the social choice could be a tie set (except at stage 2). However, what we are introducing out of necessity is a further rational constraint which is consistent for all m and which is in addition to Arrow's constraints and, as we have shown in our previous paper, all of Arrow's constraints are satisfied by our algorithm.

We take as the range of the SWF the *power set* (Stoll[1979]) of the set of all possible rankings, ρ(A), where

$$A = \{R^1, R^2, \ldots, R^q\}$$

The set $\{R^1, R^2, ..., R^q\}$ represents every possible ranking of the alternatives a,b,c... . $q = m!$. ρ(A) is the set of all possible subsets of A. If the vote is split precisely equally among every possible ranking, then the social choice would be equal to the tie set A. If there is a singular solution, then the social choice is equal to one of the elements of the set A. Other subsets of A would represent tie solutions of varying degrees.

An Example of the Algorithm for m = 5

As an example we consider the following case:

aRb,	**aRc,**	**aTd, aRe**	
	bRc,	**bRd,**	**bRe**
		cRd,	**cTe**
			dRe

From the stage 2 solutions given above, we can generate the stage 3 and stage 4 solutions. We present the stage 4 solutions below and demonstrate how to construct the stage 5 solution from them. In order to simplify the notation, we introduce a shorthand for the R and T operators as follows: aRbRc...yRz becomes **abc...yz** and aTb becomes **(a,b)** so that, for example, aRbTcRdRe becomes **a(b,c)de**.

The stage 4 solutions form a matrix as follows:

Stage 4 Letter Combinations		Stage 4 Winning Matrix
(1)	**a,b,c,d**	**abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b**
(2)	**a,b,c,e**	**ab(c,e)**
(3)	**a,b,d,e**	**abde, b(a,d)e, (a,d)be**
(4)	**a,c,d,e**	**acde, c(a,d)e, (a,d)(c,e), a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e)**
(5)	**b,c,d,e**	**bcde, bd(c,e), b(c,e)d**

The SWF algorithm proceeds as follows. Consider each letter combination in the matrix, M(j,k), in lexicographical order that has not been covered at least once i.e. M(1,1) through M(5,3). Find a stage 5 element which covers each such that no stage 4 element is covered more than twice and such that upon reduction from stage 5 to stage 4 there are no elements in the reduced set that are not part of the stage 4 solution that are covered more than once. When each element has been considered, go over the matrix again in lexicographical order and cover again those elements that have only been covered once. A stage 5 element "covers" a stage 4 element if a letter can be blotted out of the stage 5 element and the resultant element is identical to the stage 4 element.

1) Consider M (1,1) = **abcd**. Insert an **e** in every possible position (starting to the right and working to the left) and compute the rating which is the number of stage 4 winners covered as follows:

Stage 4 Element	Potential Element of Stage 5 Winning Set		Rating
M(1,1) = **abcd**	**abcde**	4	
	abced	1	
	abecd	1	
	aebcd	1	
	eabcd	1	
	abc(d,e)		1
	ab(c,e)d		2
	a(b,e)cd		1

(a,e)bcd	1

2) Pick highest rated one: **abcde**. Update list of covered elements. The number of times the element has been covered is indicated in parentheses.

Covering Element	Covered Elements
abcde	**abcd** (1), **abde** (1), **acde** (1), **bcde** (1)

3) Check to see that no element is covered more than twice. If it is, don't add covering element to winning set and go back to (1). If it is not, go to (4).

4) Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

(1) Blot out an **e**: We get **abcd**. **abcd** in winning set.
(2) Blot out a **d**: We get **abce**. **abce** not in winning set, but covered only once.
(3) Blot out a **c**: We get **abde**. **abde** in winning set.
(4) Blot out a **b**: We get **acde**. **acde** in winning set.
(5) Blot out an **a**: We get **bcde**. **bcde** in winning set.

Winning set is now {**abcde**}.
Element not in stage 4 winning matrix, but covered only once: **abce**

5) Go back to (1) and proceed with next element.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,2) = **acbd**	**acbde**	3
	acbed	1
	acebd	1
	aecbd	1
	eacbd	1
	acb(d,e)	1
	ac(b,e)d	1

a(c,e)bd	1
(a,e)cbd	1

Pick highest rated one: **acbde**. Update list of covered elements.

Covering Element	Covered Elements
acbde	**abcd** (1), **abde** (2), **acde** (2), **bcde** (1) **acbd** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde**}

(1) Blot out an **e**: We get **abcd**, **acbd**. In winning set: **abcd**, **abdc.**
(2) Blot out a **d**: We get **abce**, **acbe**. Not in winning set: **abce**, **acbe**.
(3) Blot out a **c**: We get **abde**, **abde**. In winning set: **abde**, **abde.**
(4) Blot out a **b**: We get **acde**, **acde**. In winning set: **acde**, **acde.**
(5) Blot out an **a**: We get **bcde**, **cbde**. In winning set: **bcde.** Not in winning set: **cbde**

Winning set is now:
{**abcde, acbde**}

Elements not in stage 4 winning matrix, but covered only once:
abce, **acbe, cbde**

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,3) = **bc(a,d)**	**bc(a,d)e**	4
	bc(a,d,e)	2
	bce(a,d)	1
	bec(a,d)	1
	ebc(a,d)	1
	b(c,e)(a,d)	3
	(b,e)c(a,d)	1

Pick highest rated one: **bc(a,d)e**. Update list of covered elements.

Covering Element	Covered Elements
bc(a,d)e	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (1), **c(a,d)e** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d)**. In winning set: **abcd, abdc, bc(a,d).**
(2) Blot out a **d**: We get **abce, acbe, bcae.** Not in winning set: **abce, acbe, bcae.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e**. In winning set: **abde, abde, b(a,d)e.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e.** In winning set: **acde, acde, c(a,d)e.**
(5) Blot out an **a**: We get **bcde, cbde, bcde**. In winning set: **bcde, bcde.** Not in winning set: **cbde**

Winning set is now:
{abcde, acbde, bc(a,d)e}

Elements not in stage 4 winning matrix, but covered only once:

abce, acbe, bcae, cbde

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,5) = **(a,d)bc**	**(a,d)bce**	2
	(a,d)bec	2
	(a,d)ebc	1
	(a,d,e)bc	2
	e(a,d)bc	1
	(a,d)b(c,e)	4
	(a,d)(b,e)c	1

Pick highest rated one: **(a,d)b(c,e)**. Update list of covered elements.

Covering Element	Covered Elements
(a,d)b(c,e)	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1) **bc(a,d)** (1), **b(a,d)e** (1), **c(a,d)e** (1), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (1), **(a,d)(c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e)**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc.**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e).** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be**. In winning set: **abde, abde, b(a,d)e, (a,d)be.**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e).**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e)**. In winning set: **bcde, bcde.** Not in winning set: **cbde, db(c,e)**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e)}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, cbde, db(c,e)

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,5) = **b(a,d)c**	**b(a,d)ce**	2
	b(a,d)ec	2
	b(a,d,e)c	2
	be(a,d)c	1
	eb(a,d)c	1
	b(a,d)(c,e)	4
	(b,e)(a,d)c	1

Pick highest rated one: **b(a,d)(c,e)**. Update list of covered elements.

Covering Element	Covered Elements
b(a,d)(c,e)	**abcd** (1), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (1), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (1) **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e)**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c.**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e).** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e).**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e.**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e).**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e)**. In winning set: **bcde, bcde, bd(c,e).** Not in winning set: **cbde, db(c,e)**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e)}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cbde, db(c,e)

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,6) = **c(a,d)b**	**c(a,d)be**	3
	c(a,d)eb	2
	c(a,d,e)b	1
	ce(a,d)b	1
	ec(a,d)b	1
	c(a,d)(b,e)	2
	(c,e)(a,d)b	2

Pick highest rated one: **c(a,d)be**. Update list of covered elements.

Covering Element	Covered Elements
c(a,d)be	**abcd** (1), **abde** (2), **acde** (2), **bcde**(2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (1), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{**abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be**}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe.** In winning set: **ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e.**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe**. In winning set: **bcde, bcde, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, cbde, db(c,e), cdbe.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(2,1) = **ab(c,e)**	**ab(c,e)d**	4
	ab(c,d,e)	1
	abd(c,e)	3
	adb(c,e)	1
	dab(c,e)	1
	a(b,d)(c,e)	1
	(a,d)b(c,e)	4

Pick highest rated one: **ab(c,e)d**. Update list of covered elements.

Covering Element	Covered Elements
ab(c,e)d	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (1), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (1), **b(c,e)d** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d}

> (1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd.**
> (2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe.**
> (3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed**
> (4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d.**
> (5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, abed, cbde, db(c,e), cdbe.

The next stage 4 winner that has not already been covered twice is **a(c,e)d.**

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating

M(4,4) = **a(c,e)d**	**a(c,e)db**	1
	a(c,e)bd	2
	a(b,c,e)d	1
	ab(c,e)d	4
	ba(c,e)d	2
	a(c,e)(b,d)	1
	(a,b)(c,e)d	2

ab(c,e)d doesn't work since **abcd** has already been covered twice. Try **a(c,e)bd**. Update list of covered elements.

Covering Element	Covered Elements
a(c,e)bd	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (1), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d.**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, abed, aebd, cbde, db(c,e), cdbe, (c,e)bd.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,5) = **(c,e)(a,d)**	**(c,e)(a,d)b**	2
	(c,e)(a,b,d)	1
	(c,e)b(a,d)	1
	(b,c,e)(a,d)	1
	b(c,e)(a,d)	3

Pick highest rated one: **b(c,e)(a,d)**

Covering Element	Covered Elements
b(c,e)(a,d)	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (1), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d).**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd.**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d)}

Elements not in stage 4 winning matrix, but covered only once:

abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, abed, aebd, be(a,d), cbde, db(c,e), cdbe, (c,e)bd.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,6) = **(a,d,e)c**	**(a,d,e)cb**	1
	(a,d,e)bc	2
	(a,b,d,e)c	1
	b(a,d,e)c	2
	(a,d,e)(b,c)	1

Pick highest rated one: **(a,d,e)bc**

Covering Element	Covered Elements
(a,d,e)bc	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (1), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c.**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d.** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc}

Elements not in stage 4 winning matrix, but covered only once:
abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, abed, aebd, be(a,d), (a,d,e)b, cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,7) = **d(a,c,e)**	**d(a,c,e)b**	1
	d(a,b,c,e)	1
	db(a,c,e)	1
	bd(a,c,e)	2
	(b,d)(a,c,e)	1

Pick highest rated one: **bd(a,c,e)**

Covering Element	Covered Elements
bd(a,c,e)	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (1), **bd(c,e)** (2), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc.** Not in winning set: **bd(a,c).**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e).**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e).**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e)**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e)}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), abed, aebd, be(a,d), (a,d,e)b, bd(a,e), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc.

Now we go back and make a second pass over the stage 4 elements covering those that have only been covered once again.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,3) = **b(a,d)c**	**b(a,d)ce**	2
	b(a,d)ec	2
	b(a,d,e)c	2
	be(a,d)c	1
	eb(a,d)c	1
	b(a,d)(c,e)	4
	(b,e)(a,d)c	1

Next 1-coverer in lex order: **be(a,d)c.** This does not work since **be(a,d)** has already been covered once. Try **eb(a,d)c.**

Covering Element	Covered Elements
eb(a,d)c	**abcd** (2), **abde**(2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (1), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c.** Not in winning set: **bd(a,c).**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac.**
(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d).**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)c, cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc.

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(1,6) = **c(a,d)b**	**c(a,d)be**	3
	c(a,d)eb	2
	c(a,d,e)b	1
	ce(a,d)b	1
	ec(a,d)b	1

c(a,d)(b,e)	2
(c,e)(a,d)b	2

Next 1-coverer in lex order: **c(a,d,e)b.** This does not work since **(a,d,e)b** has already been covered once. Try **ce(a,d)b.**

<u>Covering Element</u>	<u>Covered Elements</u>
ce(a,d)b	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (1), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b.**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c, ce(a,d)**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,5) = **(c,e)(a,d)**	**(c,e)(a,d)b**	2
	(c,e)(a,b,d)	1
	(c,e)b(a,d)	1
	(b,c,e)(a,d)	1
	b(c,e)(a,d)	3

Next 1-coverer in lex order: **(c,e)(a,b,d).**

Covering Element	Covered Elements
(c,e)(a,b,d)	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (1), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d)}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b)**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b).**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b).** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b).**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b)**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d).** Not in winning set: **e(a,d)c, ce(a,d)**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b)**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b).**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b)}

Elements not in stage 4 winning matrix, but covered only once:

bd(a,c), c(a,d,b), abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b)

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating

M(4,6) = **(a,d,e)c**	**(a,d,e)cb**	1	
	(a,d,e)bc		2
	(a,b,d,e)c		1
	b(a,d,e)c		2
	(a,d,e)(b,c)		1

Next 1-coverer in lex order: **(a,d,e)cb.** This doesn't work since **(a,d,e)b** has already been covered once. Try: **(a,b,d,e)c.**

<u>Covering Element</u>	<u>Covered Elements</u>
(a,b,d,e)c	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (2), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d), (a,b,d,e)c}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b), (a,b,d)c**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b), (a,b,d)c.**

(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e)**. In winning set: **abde,**

abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be. Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e).**

(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d), (a,d,e)c.** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d), (a,d,e)c.** Not in winning set: **e(a,d)c, ce(a,d)**

(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b), (b,d,e)c**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c.**

Winning set is now:

{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b), (a,b,d,e)c}

Elements not in stage 4 winning matrix, but covered only once:

bd(a,c), c(a,d,b), (a,b,d)c, abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c

Next Stage 4 Winner	Potential Element of Stage 5 Winning Set	Rating
M(4,7) = **d(a,c,e)**	**d(a,c,e)b**	1
	d(a,b,c,e)	1
	db(a,c,e)	1
	bd(a,c,e)	2
	(b,d)(a,c,e)	1

Next 1-coverer in lex order: **d(a,c,e)b.**

Covering Element	Covered Elements
d(a,c,e)b	**abcd** (2), **abde** (2), **acde** (2), **bcde** (2), **acbd** (2), **bc(a,d)** (2), **b(a,d)e** (2), **c(a,d)e** (2), **ab(c,e)** (2), **(a,d)bc** (2), **(a,d)be** (2), **(a,d)(c,e)** (2), **b(a,d)c** (2), **bd(c,e)** (2), **c(a,d)b** (2), **a(c,e)d** (2), **b(c,e)d** (2), **(c,e)(a,d)** (2), **(a,d,e)c** (2), **d(a,c,e)** (1)

Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.

Potential winning set:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,b,d), (a,b,d,e)c, d(a,c,e)b}

(1) Blot out an **e**: We get **abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, bd(a,c), b(a,d)c, c(a,d)b, c(a,d,b), (a,b,d)c, d(a,c)b**. In winning set: **abcd, abdc, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b, abcd, acbd, bc(a,d), (a,d)bc, b(a,d)c, c(a,d)b.** Not in winning set: **bd(a,c), c(a,d,b), (a,b,d)c, d(a,c)b.**
(2) Blot out a **d**: We get **abce, acbe, bcae, ab(c,e), ba(c,e), cabe, ab(c,e), a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b.** In winning set: **ab(c,e), ab(c,e).** Not in winning set: **abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b.**

(3) Blot out a **c**: We get **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b**. In winning set: **abde, abde, b(a,d)e, (a,d)be, b(a,d)e, (a,d)be.** Not in winning set: **abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b.**
(4) Blot out a **b**: We get **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), e(a,d)c, ce(a,d), (c,e)(a,d), (a,d,e)c, d(a,c,e).** In winning set: **acde, acde, c(a,d)e, (a,d)(c,e), (a,d)(c,e), c(a,d)e, a(c,e)d, a(c,e)d, (c,e)(a,d), (a,d,e)c, d(a,c,e), (c,e)(a,d), (a,d,e)c, d(a,c,e).** Not in winning set: **e(a,d)c, ce(a,d)**
(5) Blot out an **a**: We get **bcde, cbde, bcde, db(c,e), bd(c,e), cdbe, b(c,e)d, (c,e)bd, b(c,e)d, (d,e)bc, bd(c,e), ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b**. In winning set: **bcde, bcde, bd(c,e), b(c,e)d, b(c,e)d, bd(c,e).** Not in winning set: **cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b.**

Winning set is now:
{abcde, acbde, bc(a,d)e, (a,d)b(c,e), b(a,d)(c,e), c(a,d)be, ab(c,e)d, a(c,e)bd, b(c,e)(a,d), (a,d,e)bc, bd(a,c,e), eb(a,d)c, ce(a,d)b, (c,e)(a,d,b), (a,b,d,e)c, d(a,c,e)b}

Elements not in stage 4 winning matrix, but covered only once:
bd(a,c), c(a,d,b), (a,b,d)c, d(a,c)b, abce, acbe, bcae, ba(c,e), cabe, a(c,e)b, b(c,e)a, (a,e)bc, b(a,c,e), ebac, ceab, (c,e)(a,b), (a,b,e)c, (a,c,e)b, abed, aebd, be(a,d), (a,d,e)b, bd(a,e), eb(a,d), e(a,d)b, e(a,d,b), (a,b,d,e), d(a,e)b, e(a,d)c, ce(a,d), cbde, db(c,e), cdbe, (c,e)bd, (d,e)bc, ebdc, cedb, (c,e)(d,b), (b,d,e)c, d(c,e)b

This completes the solution for stage 5.

Explication of the Algorithm

The following is a formal delineation of the steps involved in the algorithm. We assume we have the correct solutions for the $(m-1)^{th}$ stage
$(m > 2)$ and want to develop the solution for the m^{th} stage.

1) Label all the alternatives alphanumerically such as **a, b, c** etc.
2) List all the $(m-1)^{th}$ stage letter combinations in lexicographical order.
3) For each $(m-1)^{th}$ stage letter combination, list the $(m-1)^{th}$ stage solution next to it forming the $(m-1)^{th}$ stage winning matrix. The elements of each solution are written, for example, **ab(c,d)** etc.
4) Consider each element in the winning matrix in lexicographical order i.e. from left to right columnwise and from top to bottom rowwise.
5) For each element in order list the possible m^{th} stage elements by inserting the remaining letter at the end of the element to form the first m^{th} stage element and then moving that letter one place to the left to form the next element etc. This represents the lexicographical ordering of the m^{th} stage elements. After this process has been completed, the remaining letter is inserted in the same way from right to left again forming elements with possible tie alternatives.
6) For each possible m^{th} stage element assign a rating which is computed by calculating the number of $(m-1)^{th}$ stage elements that are "covered" by this element where "covered" has been defined previously.
7) Choose that m^{th} stage element with the highest rating as a potential element of the m^{th} stage winning set. If there is a tie in the ratings consider the first element of the tie in lexicographical order.
8) Keep a list of the $(m-1)^{th}$ stage elements that are covered as they occur as a result of the inclusion of a potential m^{th} stage element in the winning set. For each "covered" element, keep a record as to how many times it has been covered.
8) Make sure that, when the potential element are considered to be part of the winning solution, no $(m-1)^{th}$ stage element is covered more than twice.
9) If a potential stage m element results in a $(m-1)^{th}$ stage element being covered more than twice, then consider the next element in lexicographical order of the same rating or next lower rating. Go back to step 1.

10) Check to see that, upon reducing winning set from stage 5 to stage 4, there are no elements that are not in the stage 4 winning matrix that are covered more than once.
11) If there are elements not in the winning matrix that are covered more than once, the potential element must be thrown out. Consider the next element in lexicographical order of the same or next lower rating. Go back to step 1.
12) If all potential stage 5 elements have been considered and no suitable element has been found, then go back to the last element included in the winning set that could be changed in such a way as to result in the least number of changes to the winning set. Change that element to another one thus allowing one of the presently considered elements to be used in the winning set.
13) Add the potential element to the winning set.
14) Go back to step 1 and continue until every $(m-1)^{th}$ stage element has been considered.
15) If some $(m-1)^{th}$ stage elements have not been covered twice, start over considering those particular elements in lexicographical order.
16) Continue until all $(m-1)^{th}$ stage elements have been covered exactly twice.

Proof that Algorithm Works in Every Case

We do a proof by induction. We assume that the algorithm provides solutions which are correct for stage m-1. Then we prove that the solutions are correct for stage m. We also know that the algorithm provides correct solutions for stage 3 as presented previously in the companion paper, *A Social Choice Algorithm*.

Step 1:

Any two (m-1)ary solutions will reduce to the same (m-2)ary solution for the m-2 letters they have in common.

Proof

a) We have assumed that the solutions at stage m-1 are correct.
b) There are m solutions at stage m-1, one for each of m combinations of m letters taken m-1 at a time.

e.g. for m=5, the solutions are

Letter Combination	Solution
a,**b**,**c**,**d**	Z_1^1 (**abcd**), Z_2^1 (**abcd**), Z_3^1 (**abcd**)
a,**b**,**c**,**e**	Z_1^2 (**abce**), Z_2^2 (**abce**), Z_3^2 (**abce**)
a,**b**,**d**,**e**	Z_1^3 (**abde**), Z_2^3 (**abde**), Z_3^3 (**abde**)
a,**c**,**d**,**e**	Z_1^4 (**acde**), Z_2^4 (**acde**), Z_3^4 (**acde**)
b,**c**,**d**,**e**	Z_1^5 (**bcde**), Z_2^5 (**bcde**), Z_3^5 (**bcde**)

where Z_i^j (**wxyz**) is a permutation of **wxyz**.

c) We know that when a letter is "blotted out" of a (m-1)ary solution, the solution reduces to the correct (m-2)ary solution.
d) Any two (m-1)ary solutions have m-2 letters in common.
e) Therefore, if the uncommon letter is removed from each of two (m-1)ary solutions, both will reduce to the same (m-2)ary solution.

Step 2:

For any two (m-1)ary solutions, there are elements in both solutions which have m-2 letters which are the same and in the same order. In fact and by construction, there are 2n elements in each solution which have elements with the same letters in the same order where n is the number of elements in the (m-2)ary solution.

Proof

By construction

e.g. for m=6 and the following case

aRb	aRc	aRd	eRa	fRa
	bRc	bRd	bRe	fRb
		cRd	cRe	cRf
			dRe	dRf
				eRf

we have fifth stage solutions as follows:

a,b,c,d,e: **abcde**, **bcdea**, **eabcd**, **abdce**, **cbdea**, **eacbd**, **bcead**, **deabc**, **acdeb**
a,b,c,d,f: **abcdf**, **cdfab**, **fabcd**, **abdcf**, **bacdf**, **cdfba**, **dcfab**, **facbd**, **fadbc**, **fbcda**

The fourth stage solution for **a**,**b**,**c**,**d** is **abcd**.

When we reduce the above solution for **a**,**b**,**c**,**d**,**e** we get **abcde, eabcd**, and when we reduce the above solution for **a**,**b**,**c**,**d**,**f** we get **abcdf** and **fabcd**. Both solutions reduce correctly to **abcd**.

Step 3:

Any two (m-1)ary elements with (m-2) letters in common and in the same order can be covered by one m-ary element.

e.g. the two elements **abc(d,e)** and **abc(d,f)** can be covered by **abc(d,e,f)**.

Proof

Without loss of generality, let $a_1a_2\cdots a_{m-2}$ be the m-2 letters that each element has in common. Let's say that the (m-1)th letter is X for the first element and Y for the second.

Therefore, not considering ties at the binary level, we have

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-2}$$

where a_i is a distinct member of the set, {**a**,**b**,**c**···}, for $1 < i < m-1$, and $a_i = 1$ for i=1 and i=m-1.

and $$a_1a_2\cdots a_{j-1}Ya_j\cdots a_{m-2}$$

where a_j is a distinct member of the set, {**a**,**b**,**c**···}, for $1 < j < m-1$, and $a_j = 1$ for j=1 and j=m-1.

We construct the m-ary element by taking the element, $a_1a_2\cdots a_{m-2}$, and inserting X between a_{i-1} and a_i and Y between a_{j-1} and a_j as follows:

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}Ya_j\cdots a_{m-2}.$$

Clearly, if i=j, we may have either

$$a_1a_2\cdots a_{i-1}XYa_i\cdots a_{m-2}$$

or $$a_1a_2\cdots a_{i-1}YXa_i\cdots a_{m-2}.$$

When ties at the binary level are considered, we have

$a_1a_2\cdots(a_{i-1},X)a_i\cdots a_{m-2}$ or $a_1a_2\cdots(a_{i-1},X,a_i)\cdots a_{m-2}$ or $a_1a_2\cdots a_{i-1}(X,a_i)\cdots a_{m-2}$
and
$a_1a_2\cdots(a_{j-1},Y)a_j\cdots a_{m-2}$ or $a_1a_2\cdots(a_{j-1},Y,a_j)\cdots a_{m-2}$ or $a_1a_2\cdots a_{j-1}(Y,a_j)\cdots a_{m-2}$

We construct the m-ary element in the same way using parentheses as appropriate.

e.g. for

$a_1a_2\cdots(a_{i-1},X)a_i\cdots a_{m-2}$ and $a_1a_2\cdots(a_{j-1},Y)a_j\cdots a_{m-2}$

and for i=j, we have

$a_1a_2\cdots(a_{i-1},X,Y)a_j\cdots a_{m-2}$

The proof is not substantially changed if some of the alternatives are tied:

e.g. the two elements **a(b,c)de** and **a(b,c)df** can be covered by **a(b,c)def** and **a(b,d)e** and **a(c,d)e** can be covered by **a(b,c,d)e.**

Step 4:

Each element of the (m-1)ary winning matrix can combine with at least one other element of the winning matrix in such a way as to form an m-ary element that covers those elements so combined. There are enough such elements to cover all elements in the (m-1)ary winning matrix at least once. We will call such m-ary elements *primary elements.*

Proof

a) Since any two (m-1)ary rows have to reduce to the same solution at stage m-2 for the m-2 letters they have in common, they will have 2n elements in common where n is the number of elements in the particular row at stage m-2.

e.g.

row p: $X[b_1^p]^1,X[b_1^p]^2,X[b_2^p]^1,X[b_2^p]^2, \cdots,X[b_n^p]^1,X[b_n^p]^2,X[b_{n+1}^p], \cdots ,X[b_t^p]$

row q: $Y[b_1^q]^1,Y[b_1^q]^2,Y[b_2^q]^1,Y[b_2^q]^2, \cdots,Y[b_n^q]^1,Y[b_n^q]^2,Y[b_{n+1}^q], \cdots ,Y[b_s^q]$

where

row x is that row of the winning matrix whose letter combination does not include x.

b_z^p and b_z^q are (m-1)ary elements such as **badc…t**. Each element contains m-1 letters.

$b_z^p = b_z^q$ for $1 \le z \le n$.

$b_z^p \ne b_z^q$ for $n+1 \le z \le \min(t,s)$, where min(t,s) is the minimum of t and s.

X and Y are letters and the brackets represent an operator such that X[AB…P] = XAB…P or AXB…P or … or AB…XP or AB…PX.

$X[b_z^v]^1$ represents a different permutation of X and b_z^v than does $X[b_z^v]^2$

b) $X[b_z^p]^j$ and $Y[b_z^q]^j$ can be covered at stage m by $X[Y[b_i^p]]$ for $1 \le z \le n$.

c) That leaves the elements $X[b_{n+1}^p], \cdots, X[b_t^p]$ and $Y[b_{n+1}^q], \cdots, Y[b_s^q]$. By construction these elements cover (m-2)ary elements in rows other than p and q. So for each of these elements there exists an element in another row of the (m-1)ary winning matrix such that they each have m-2 letters in common and in the same order. Therefore, by **Step (3)** there is an m-ary element that covers each of these elements and at least one other.

Example

Let the (m-1)ary winning matrix be

acde	**cdea**	**eacd**
acdf	**cdfa**	**facd**
acef	**cefa**	**efac**
adef	**defa**	**efad**
cdef		

The 5-ary set {**acdef**, **cdefa**, **efacd**} consists of primary elements and covers the 4-ary winning matrix exactly once.

Definition: Interference—when there are any two elements in a potential m-ary winning set that, when reduced to the (m-1)ary level, generate the same element which is not in the (m-1)ary winning matrix.

Step 5:

Two primary elements cannot interfere with each other.

Proof:

The only way interference can occur is if there are at stage m two elements such that, when a letter is blotted out, both elements reduce to the same (m-1)ary element and this element is not in the (m-1)ary winning matrix.

Without loss of generality, let the two m-ary primary elements be

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-2} \text{ and } a_1a_2\cdots a_{j-1}Xa_j\cdots a_{m-2}$$

When X is blotted out these both reduce to

$$a_1a_2\cdots a_{m-2}$$

so that there are two such elements at stage m-1. We assume that this element is not in the winning matrix and prove the assertion that two primary elements cannot interfere by contradiction.

Because both m-ary elements under consideration are primaries, they were both formed by merging two elements from the (m-1)ary winning matrix. Let these elements be

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{i-1}Xa_i\cdots a_{m-2},\ a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{i-1}Xa_i\cdots a_{m-2}$$

and

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{j-1}Xa_j\cdots a_{m-2},\ a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{j-1}Xa_j\cdots a_{m-2},$$

respectively.

This implies that at the (m-2) stage there is an element

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

and an element

$$a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

since there are two of each of them at the (m-1)ary stage when an X is blotted out and we know, by assumption, that the (m-1)ary solution is correct. Therefore, there must be an element on row X (where row X is the row in the (m-1)ary winning matrix which does not contain an X in its letter combination), stage (m-1) that reduces to

$$a_1a_2\cdots a_{k-1}\, a_{k+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

when a K is blotted out and to

$$a_1a_2\cdots a_{l-1}\, a_{l+1}\cdots a_{i-1}a_i\cdots a_{m-2}$$

when an L is blotted out since every row at stage (m-1) must reduce correctly. The only element for which this is possible is

$$a_1a_2\cdots a_{m-2}$$

and, therefore, this element must be in the (m-1)ary winning matrix which contradicts the assumption and the assertion is proven.

Example

Consider the following stage 4 winning matrix:

abcd		
abce	**bcea**	**eabc**
abde	**bdea**	**eabd**
acde	**cdea**	**eacd**
bcde		

We can form the stage 5 primary **abcde** from **abce** and **abde** and the stage 5 primary **bcdea** from **bcea** and **bdea**, respectively. Then on blotting out an **a** at stage 5 we will have two **bcde**s. Therefore, **bcde** must be in the stage 4 winning matrix or else interference would occur. Since at stage 4 there are two **bce**s if an **a** is blotted out of row d and two **bde**s if an **a** is blotted out of row c, this implies that there is a **bcde** on row a.

Step 6:

For each (m-1)ary element there are m permutations of that element and the last remaining letter. (There are m letters altogether.) One of them is the primary element. So there are m-1 other permutations. Some of these cover two (m-1)ary elements and some cover one. We call these other permutations secondaries and we say they are related to the primary element from which they are derived.

e.g.

Let

$$a_1a_2\cdots a_{m-1}$$

be the (m-1)ary element. Then we have the possible set of m-ary permutations as follows:

$$\{X\, a_1a_2\cdots a_{m-1},\ a_1X\, a_2\cdots a_{m-1},\ \cdots,\ a_1a_2\cdots a_{m-1}X\}$$

Let

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$

be the primary element which covers two or more (m-1)ary elements. One of the covered elements is $a_1a_2\cdots a_{m-1}$ by construction.

Consider

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$

If a_i or a_{i-1} is blotted out, the resultant (m-1)ary elements might be in the (m-1)ary winning matrix. If $a_1a_2\cdots a_{i-1}Xa_{i+1}\cdots a_{m-1}$ is in the winning matrix, for example, then the element $a_1a_2\cdots a_{i-1}a_iX\cdots a_{m-1}$ covers two (m-1)ary elements. Every other permutation of X and $a_1a_2\cdots a_{m-1}$ results in an element in which the X is out of position from its place in the primary element and, hence, the resultant element can only be a 1-coverer and only when the X is blotted out. Therefore, a secondary can cover one or in two cases possibly two elements.

<u>Step 7:</u>

There are at least two secondaries derived from any given m-ary primary that will not interfere with any other m-ary primary or secondary.

<u>Proof</u>

Any primary differs from any other primary or secondary by having at least one letter in a different place. Let's consider a given primary or secondary element

$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}Ya_j\cdots a_{m-2}$

Then we consider a second primary element such as

$a_1a_2\cdots a_{k-1}Xa_k\cdots a_{l-1}Ya_l\cdots a_{m-2}$

in which the letter Y is the one letter definitely in a different position from the preceding element. This second primary is unrelated to the first element since it is not derived from it. The letter X "slides" along the second element forming different permutations at different positions, ($0 < k < m$, $a_0 = a_{m-1} = 1$), and different related secondaries. In the first element, X is fixed. Now when $k=i$, there is possible interference when a Y is blotted out since both elements reduce down to

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{j-1}a_j\cdots a_{m-2}.$$

In all other positions of X (or values of k), there are two letters out of synch for the two elements so they will not reduce down to the same (m-1)ary element and hence there will be no interference.

If X and Y are adjacent in the first element

$$a_1a_2\cdots a_{j-1}XYa_j\cdots a_{m-2}$$

then there are two positions of X which could cause interference as follows

$$a_1a_2\cdots a_{l-1}XYa_l\cdots a_{m-2},\ l=j$$

and

$$a_1a_2\cdots a_{l-1}YXa_l\cdots a_{m-2},\ l=j$$

Therefore, there are at most two positions that could cause interference between a secondary and an unrelated element (primary or secondary).

If there are ties in the first element as follows,

$$a_1a_2\cdots(a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j)\cdots a_{m-2},$$

then the second element will produce interference only for those positions inside the parentheses. All other kinds of ties (not involving X and Y tied together) do not alter the above analysis.

A secondary cannot interfere with the primary it is derived from since for the two elements

$$a_1a_2\cdots a_{i-1}Xa_i\cdots a_{m-1}$$

and

$$a_1a_2\cdots a_{k-1}Xa_k\cdots a_{m-1}$$

when a_j is blotted out for any value of j except a_j=X, the two reduced elements will not be identical since each X will be in a different position and when X is blotted out the reduced element

$$a_1a_2\cdots a_{m-2}$$

is in the stage m-1 winning matrix by definition.

Therefore, the assertion is proved true.

Step 8:

There are, therefore, m-2 other secondary elements which are non-interfering not considering ties for the moment. Choose one of these if necessary (derived from each primary) to be the second m-ary element to cover each (m-1)ary element. Each (m-1)ary element is then covered twice in such a way that, when the solution is reduced from m-ary to (m-1)ary, every other element in the reduced solution is covered at most once. Therefore, we have proven that, if there is a correct solution at stage m-1, it is possible to find a correct solution for stage m. We know all the solutions for m=3. Therefore, a solution exists for m= $4\cdots\infty$.

When ties are considered, the number of possible non-interfering elements is reduced when both X and Y are in the tie by the number of tied alternatives. At least, when every alternative is tied, there are still two non-interfering elements as shown by the following. Assume the first element is as follows:

$$(a_1,a_2,\cdots a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})$$

Then the second element would be non-interfering for the following positions of X:

$$X(a_1,a_2,\cdots a_{i-1},a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})$$

and

$$(a_1,a_2,\cdots a_{i-1},X,a_i,\cdots a_{j-1},Y,a_j,\cdots,a_{m-2})X$$

Therefore, there are at least two non-interfering elements. Since we know all the solutions for m=3, solutions exist for $m= 4\cdots\infty$.

New Directions

Since many of the solutions are ties, we may use an additional criterion to choose among them. In fact we could introduce the concept of "digital utility" which would be a measure of the "goodness of fit" of each of the elements of the tie set. We could measure for each individual voter the goodness of fit of his preference list with the social choice by measuring the "distance," for each alternative, between the position of that alternative in the voter's preference list and the position of that alternative in the social choice. For instance, if voter i places alternative **a** 2nd in his list and the social choice places **a** 4th, there is a distance of 2 between the individual choice and the social choice. Summing over all alternatives and all individuals, we could get a measure of the digital utility for each element of the tie set. The element with the lowest summation would be the one with the highest digital utility, and, therefore, could be chosen as the social choice.

There is reason to believe that the social choices produced by the algorithm we have presented are stable in that it doesn't pay for any voter to vote insincerely. We quote Murakami (1968): "Therefore, if a democracy is based on pairwise comparisons, the outcome of sincere individual decisions is, if it exists at all, always stable. Any insincere or strategic move cannot improve the situation for any individual. This is one of the essential features of democracy based on pairwise comparison. Therefore, insofar as a democracy is based on pairwise comparisons, a distinction between individual decisions and individual preferences may not be so important."

The aspect of pairwise comparisons also opens another door: that of probabilistic voting systems based upon limited information from each individual. Instead of millions of voters exhaustively ranking hundreds of alternatives, we can envision a voting system in which different voters are assigned different pairs of candidates to be ranked on a pairwise or a partial list ranking basis. Then all this information can be integrated to form the social choice with the probability of error made as low as desired by increasing the number of pairwise or partially ordered lists considered. The results could be compared with non-probabilistic voting systems for accuracy of results and effort (on the voters' parts) expended.

Conclusions

In this paper we have proven that social choice is possible according to Arrow's five criteria for any number of alternatives, any number of voters and when individual and social choices are expressed in terms of the R operator. We have demonstrated an algorithm and proven that it provides solutions for all cases. Therefore, social choice is indeed possible.

References

Arrow, Kenneth J. (1951) *Social Choice and Individual Values* (John Wiley & Sons Inc., New York).

Condorcet, M. J. A. N., marquis de (1785) *Essai sur l'application de l'analyse á la probabilité des décisions rendues á la pluralité des voix* (Imprimerie Royale, Paris).

Lawrence, John C., (1997) A Social Choice Algorithm

Murakami, Y. (1968) *Logic and Social Choice* (Routledge & Kegan Paul Ltd., London) 79.

Stoll, Robert R., (1979) *Set Theory and Logic* (Dover Publications Inc., New York).