Fourth LACCEI International Latin American and Caribbean Conference on Engineering and Technology (LACCEI '2006)
"Breaking Frontiers and Barriers in Engineering: Education, Research and Practice", 21-23 June 2006, Mayaguez, Puerto Rico

Systems of Inequations

Jeffrey L. Duffany, Ph.D.
Universidad del Turabo
Gurabo, PR USA
jduffany@suagm.edu

Abstract

A system of inequations is equivalent to the compatibility problem which is one of the best known and classic examples of np-complete problem. A solution method is described which is analogous to solving systems of equations using substitution and elimination of variables. A decision function is used to determine which variables to combine. The solution method can be implemented using set-theoretical operators such as union and intersection or using a combination of modular and traditional arithmetic operators. The performance of the basic method is shown to be significantly improved when used in a variation called the equivalence class subset algorithm.

1 Introduction

A system of inequations is represented by a set of n variables x_i and a set of compatibilities involving all pairs of variables (x_i, x_j). For example suppose n=2 and the system is represented by inequation (1):

$$x_1 \neq x_2 \tag{1}$$

For this example $x_1 = 1$ and $x_2 = 2$ is a solution as is $x_1 = p$ and $x_2 = np$, since both satisfy every inequation in the system. In general, the solution to a system of inequations is drawn from an arbitrary set of distinct elements. There are an infinite number of solutions to this system but a solution s* is called optimal only if the number of elements in the solution set is minimum over all possible solution sets {s}. In this case the cardinality of the optimal solution k*=max(s*) is 2 since s*={1,2} is a solution and it is not possible to solve this system with a set of lower cardinality. The cardinality of a *solution* k=max(s) is not the same as the cardinality of the *solution* |s| which is always equal to the dimension of the system (i.e., the number of variables "n"). A system of inequations can be represented by a binary symmetric square matrix A, with a zero representing a compatibility and a one an incompatibility as in equation (2).

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

The matrix A in equation (2) is a representation of the system of inequations $x_1 \neq x_2$. There are always two ones in the A matrix for every inequation representing both $x_i \neq x_j$ and $x_j \neq x_i$. A system of inequations can also be represented by inequation (3):

$$Ax \neq x \tag{3}$$

if the addition operator "+" in the vector product is replaced by the union ($\cup$) set operator. The variable x_1 is represented as the first row of column vector x and x_2 is represented by the second row of column vector x. Analogous to a matrix representation of equations, inequation (3) represents a system of inequations of the form $x_i \neq x_j \cup x_k \ldots$. Note that "$\neq$"in inequation (3) is interpreted differently from standard usage when either side of the expression is a set of variables. In this case the "$\neq$" symbol distributes over each element of the set. In other words, $x_i \neq x_j \cup x_k$ is equivalent to inequations $x_i \neq x_j$ and $x_i \neq x_k$. Another way to express this would be using the set-theoretical $\notin$ (not a member of) symbol. However the $\neq$ symbol will be used to emphasize the similarity between systems of equations and systems of inequations. Systems of inequations are equivalent to the well-known compatibility problem[Clay,2005] which is categorized as NP-hard[Weisstein, 2005a][Horowitz, 1978].

2 Substitution and Elimination of Variables

A solution technique for systems of inequations can be defined which is analogous to the technique used for systems of equations, also known as the method of substitution and elimination of variables. This method always finds a feasible solution "s" but the solution that it finds may or may not be optimal. An example of substitution of variables is given by the following pair of inequations (4) where variable x_1 is used to substitute for and eliminate variable x_4:

$$\begin{matrix} x_1 \neq x_2 \cup x_3 \\ x_4 \neq x_3 \cup x_5 \end{matrix} \qquad \Rightarrow \qquad x_1 \neq x_2 \cup x_3 \cup x_5 \tag{4}$$

Since there is no constraint that $x_1 \neq x_4$ in the pair of inequations (4) it is possible to set $x_1 = x_1 \cup x_4$ and eliminate x_4 (the symbol "$\cup$" representing the union operator). The union operator can be considered the set-theoretical equivalent of the addition operator. The combined inequations assert that $x_1 \neq x_2$ and that $x_1 \neq x_3$ and that $x_1 \neq x_5$. An optimal solution is given by s*=(1,2,2,1,2). The substitution of x_1 for x_4 results in $x_1=x_4=1$ and $x_2=x_3=x_5=2$ in the optimal solution. The variables have been partitioned into two sets $\{x_1,x_4\}$ and $\{x_2,x_3,x_5\}$. Since k*=2 any optimal solution will partition the variables into two sets each of which is referred to as an *equivalence class* or *independent set*. This leads to the following observation.

Observation: *For any system of inequations* a*ny two variables that are in the same equivalence class of any optimal solution can be associated to create a new system of inequations which has the same optimal solution cardinality k* as the original system.*

Combining variables in this fashion will always lead to a feasible but not necessarily optimal solution. This process can be continued until eventually $x_i \neq x_j$ for all pairs of variables (x_i, x_j). At this point either an optimal solution s* has been found or a feasible but not optimal solution s(k>k*) has been found. Back substitution is used to find the solution for the original system.

For every system of n inequations there is at least one optimal solution of minimum cardinality k* and exactly one trivial solution s(n)={1,2,3…n}. The trivial solution s(n) represents the association of each variable to a different integer $x_i = i$, i$\in$ {1,2,3,..n}. In the system $x_1 \neq x_2$ the unique optimal solution s* and the trivial solution s(n) are the same s*=s(n)=(1,2). It should also be noted that in this case the number of solutions of cardinality less than k*=2 and greater than n=2 is zero. In general, any solution that is not optimal has cardinality k>k* and can be referred to as a suboptimal solution. For any given system there can be a large number of suboptimal solutions for k*<k< n.

The number of pairs of variables that can be substituted for one another in a given system is an integer between 0 and n(n-1)/2 inclusive. Finding an optimal solution for a system of inequations is equivalent to having a decision function that can for any A matrix select two variables that are in the same equivalence class of some optimal solution s* in {s*}. For any system of inequations the set X of off-diagonal zeros of the matrix A represent the set of all pairs of variables that can be combined into equivalence classes $X=\{xi,xj \mid a_{ij}=0, i \neq j\}$. If |X|=0 ($X=\{\phi\}$) then the system is called a complete system. For any system that is not complete there is always at least one pair of variables that when combined leads to an optimal solution. If |X|=n(n-1)/2 then all variables are in the same equivalence class and $X_{ec}=X$, where X_{ec} is the subset of X corresponding to an optimal or suboptimal solution "s" $X_{ec}=\{xi,xj|xi,xj \text{ in } s\}$.

Finding an optimal solution for a system of inequations can be accomplished by finding a mapping function f(X) which partitions the set X= {xi,xj | aij=0} into two mutually exclusive subsets $X_a \cup X_c=X$, $X_a \cap X_c=\{\phi\}$. The set X_c represents the subset of X for which the variables xi and xj can be found in the same equivalence class of at least one optimal solution. The set X_a represents the subset of X for which the variables (xi,xj) are never found in the same equivalence class of any optimal solution. The subscripts *c* and *a* of X_a and X_c are abbreviations of the words chromatic and achromatic. If two variables are combined and the cardinality of an optimal solution of the new A' matrix is the same as the original A matrix then the operation it can be said to represent a *chromatic* transformation. If not then it can be said to be an *achromatic* transformation. Without loss of generality the mapping function f(X) can be referred to as the decision function f(A). For example the decision function f(A) could take every $x_{ij} \in X$ and map each x_{ij} into a 1 or a 0, depending on whether association will lead to an optimal solution or not.

3 Canonical Form

Consider a system of inequations in 5 variables as shown in equation 5(a). In a manner similar to what was done for equation (4) an optimal solution is easily found as s*=(1,2,1,2,1). There is only one unique optimal solution s* which partitions the variables into two equivalence classes $\{x_1,x_3,x_5\}$ and $\{x_2,x_4\}$.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \text{(a)} \qquad A^* = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \text{(b)} \qquad (5)$$

The matrix A* given in equation 5(b) is a permutation of the variables of A that corresponds to the partitioning into the equivalence classes of s*. Note that A^* is in block diagonal form with square submatrices of zeros of dimension 3x3 and 2x2 along the main diagonal. The matrix A^* resembles the echelon form used in solving systems of equations. It can be regarded as a canonical form in that any system of inequations can always be represented in this form. The canonical form can also be regarded as a spectral decomposition of the matrix A by a permutation matrix P as in equation (6):

$$A^* = P^{-1}AP \qquad (6)$$

If the variables are ordered such that the last one of each equivalence class cannot combine with the first variable of the next, the canonical form A* has the property that an optimal solution s* can be determined by repeatedly choosing and associating (i.e., combining) variables whose subscript differs by one (x_i,x_{i+1}) for $i \in \{1,2,...n-1\}$. In example 5(b) this would require switching the last two rows. Using this technique the canonical form A* can be used to uniquely represent an optimal solution vector s*.

From Equation 5(a) it is seen that the off-diagonal zeros represent the set X and that there are six ways of choosing pairs of the 5 variables xij, ij$\in$ 1,2,..5 ,i$\neq$j, for the purpose of substitution and elimination. Of the six exactly four belong to $X_c=\{x_{13},x_{15},x_{35},x_{24}\}$ and exactly two belong to $X_a=\{x_{14},x_{25}\}$. In equation 5(b) the set X_c corresponds to zeros inside the diagonal blocks and the set X_a corresponds to zeros outside the diagonal blocks. In general systems that have more than one solution will have members of X_c outside of the diagonal blocks.

4 K-partite Systems

The matrix A* in Figure 5(b) is an incomplete bipartite system due to the zeros outside of the diagonal blocks. A complete bipartite system has all ones outside of the diagonal blocks. Similarly, a complete k-partite system has a unique optimal solution s* which partitions the variables into k* equivalence classes and when permuted into canonical form has all ones outside the diagonal blocks. A complete k-partite system is represented by the symbol $K_{p1p2..pk}$ where the values p_i represent the number of variables in each equivalence class. This is illustrated in equation (7) for the complete k-partite system K_{322}.

$$K_{322} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{7}$$

The system K_{322} is an overconstrained system. Removing any single constraint will create a different system K'_{322} which has the same unique optimal solution as the original system. In general more than one constraint can be removed from a complete k-partite system without changing the optimal solution s*.

Let A be any system of inequations with an optimal solution vector s*. Let the cardinality of the equivalence classes of s* be given by $(p_1,p_2 \ldots p_k)$. Then there is a complete k-partite system $K_{p1p2..pk}$ that has the same optimal solution vector s* as A which can be derived from A by changing all zeros to ones everywhere outside the block diagonals of the canonical form of the A matrix. For example refer to equation 5(b) which has two zeros outside the block diagonal. Either or both of these zeros can be changed to ones creating a new and more overconstrained system of inequations with the same unique optimal solution vector s*. If both zeros are changed to one it results in the complete bipartite system K_{32} having the same optimal solution vector s*=(1,1,1,2,2).

5 Decision Functions

A decision function f(A) is a function which imposes an ordering on the set X of pairs of variables that reflects the likelihood they belong to the same equivalence class of an optimal solution. For example the decision function f(A) = A for a system of inequations can be considered an ordering of the set X where each pair of variables is given the same likelihood or probability of being in the same equivalence class since for f(A) = A, aij=0 for all xij in X. Since this decision function provides no information regarding the ordering of the pairs it implies that to find an optimal solution all of the possibilities must be searched. The decision function f(A) = A can be considered an ambivalent or neutral decision function.

On the opposite extreme is the perfect decision function which will always choose two variables (xi,xj) of a system of inequations that are in the same equivalence class of an optimal solution s*. However even if a perfect decision function does not exist it may be possible to find an imperfect decision function such that a particular xij∈ X is more likely than another to be in the same equivalence class of some s* in{s*}. The existence of even an imperfect decision function could imply the existence of an algorithm for finding an optimal solution without having to search all of the possibilities.

The canonical form shown in equations 5(b) and (7) represent the variables of a system partitioned into equivalence classes. In particular, it is seen that in any complete k-partite system such as K_{322} the rows of the matrix are identical for each variable in any equivalence class. If each row is considered as a vector each pair of variables in the same equivalence class share the same set of constraints. In other words the intersection (logical AND) of the two sets of constraints is the same for all pairs of variables in the same equivalence class. This coincidentally is equal to the inner product of the two vectors representing the rows of the A matrix as indicated in equation (8).

$$xi \bullet xj = | xi \cap xj | \tag{8}$$

To calculate the intersection of the set of constraints of every variable with every other variable in a system of inequations matrix multiplication can be used. In other words equation (8) can be generalized for every pair of variables in the system by using the square of the A matrix (A^2). For example equation (9(b)) shows the result of applying equation (8) to all pairs of variables (xi,xj) in the complete k-partite system K_{322} of Equation (7) by squaring the matrix.

$$K_{322} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 \end{bmatrix} \text{(a)} \qquad K^2_{322} = \begin{bmatrix} 4 & 4 & 4 & 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 2 & 2 & 2 & 2 \\ 4 & 4 & 4 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 5 & 5 & 3 & 3 \\ 2 & 2 & 2 & 5 & 5 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 5 & 5 \\ 2 & 2 & 2 & 3 & 3 & 5 & 5 \end{bmatrix} \text{(b)} \tag{9}$$

The value of K^2_{322} for all pairs within an equivalence class is either 4 or 5 and that this value is greater than that of any pair chosen from two different equivalence classes. The decision function f(A) = max(A^2) is a perfect decision function for all complete k-partite systems regardless of optimal solution cardinality k* or system dimension n {n| n>0, n->∞ }. For systems which are not complete k-partite the pair of variables (xi,xj) corresponding to the maximum value f(A) = max(A^2) is not guaranteed to be in the same equivalence class of an optimal solution {s*}. Note that the decision function f(A)=max(A^2) is global in the sense that it makes use of information about every possible choice of variables {(xi,xj) | (i,j)∈ {1,2,3....n}} when making a decision. A geometrical interpretation of the decision function f(A)=max(A^2) is that it chooses two points that are as close together as possible without touching.

For another example of the decision function f(A) = max(A^2) consider a system of n=100 variables generated at random with solution cardinality k*= max(s*) = 3. The matrix A for this system was squared and the intersection between every (xi,xj) pair in X is shown in Figure 1. The solid line represents the distribution of intersection between pairs of variables in the same equivalence class X_{ec} as determined from the canonical form of the A matrix corresponding to s*. The dashed line in Figure 1 represents the distribution of the intersection between pairs of variables in different equivalence classes $\overline{X}_{ec}$.

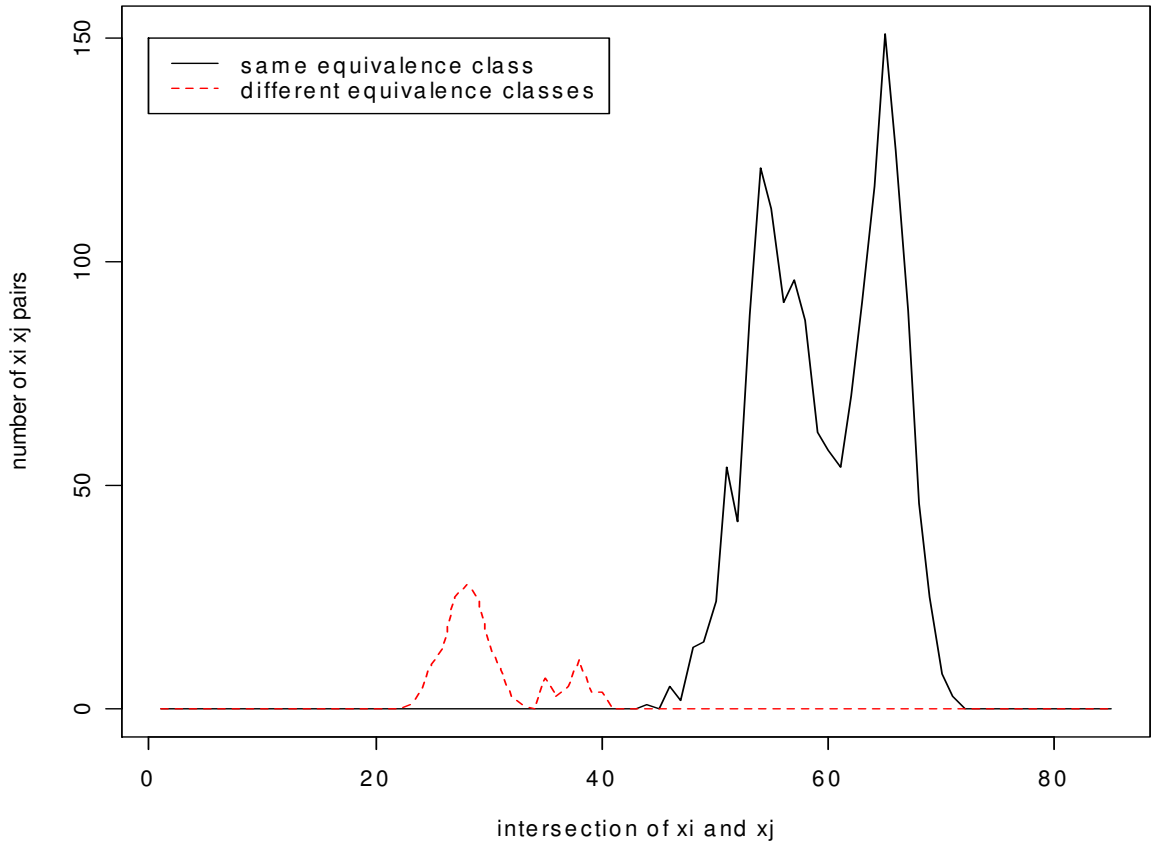

Figure 1. Result of applying $f(A) = A^2$ to a random system of 100 variables

Figure 1 corresponds to a matrix of dimension n=100 that has ten thousand aij values (3740 zeros and 7260 ones). Of the 3740 zeros 100 are on the main diagonal leaving a total of |X| = 3640 off of the main diagonal. Of the 3640 a total of X_{ec} = 3302 were inside a diagonal block and $\overline{X}_{ec}$ = 338 were outside any diagonal block of the canonical form of A corresponding to an optimal solution vector s*. Even though the A matrix was not complete k-partite it was close enough to be in the region of convergence of the decision function $f(A) = \max(A^2)$ since it separated X into two sets X_{ec} and $\overline{X}_{ec}$ whose ranges did not overlap. In other words the intersection of any pair of variables in the same equivalence class of s* was greater than that of any pair of variables in different equivalence classes of s*. Note that sets X_{ec} and $\overline{X}_{ec}$ are not equivalent to X_a and X_c. All x_{ij} in X_{ec} of s* are in X_c while x_{ij} in $\overline{X}_{ec}$ may be in either X_a or X_c.

In general the perfect separation of distributions in Figure 1 will not be the case for all systems of inequations. In some cases there will be a perfect separation or partial overlap but in other cases there will be a complete overlap of the distributions. To make a correct decision it is not necessary for all the values of A^2 within an equivalence class to be greater than every A^2 value for variables in different equivalence classes. To make a correct decision it is necessary that one pair of variables in an equivalence class of an optimal solution s* have an intersection greater than the largest intersection of any two variables that are not in the same equivalence class of any optimal solution {s*}.

6 Algorithm

Using a decision function f(A) an algorithm can be derived for solving systems of inequations (Figure 2). The algorithm is recursive. It takes as input an nxn matrix A and reduces it to an (n-1)x(n-1) system, calling itself to solve the reduced system. The solution cardinality k is the dimension of the reduced A matrix when no more variables can be combined.

ineq(A)
ij ⇔f(A)
if {ij}={ϕ} return A
xi=xi ∪ xj
A=A-xj
ineq(A)

Figure 2. Algorithm for solving systems of inequations

The decision function f(A) identifies a pair of variables x_i and x_j (also represented as ij or xij) that are likely to be in the same equivalence class. If using f(A) = max(A^2) then a given pair would be chosen because the intersection of their constraints was greater than or equal to that of any other pair of variables. The two variables xi and xj are combined using the set union operation (logical OR or modulo 2 addition) and then row i and column i of the A matrix are updated to the "sum" or union of xi and xj. The binary subtraction operator (-) is used to represent the elimination of the variable xj from the system (i.e., the removal of row j and column j from A). The algorithm terminates when the set {ij} is empty returning the same A matrix that it received as input with the solution indicated by rows and columns permuted into canonical form. The algorithm could also return the solution vector s as shown in Figure 3.

ineq(A)
s={1,2,3,...n}
ij ⇔ max(A^2)
if ij ={ϕ} return s
xi=xi ∪ xj
A=A-xj
s[s=j]=i
s[s>j]=s[s>j]-1
ineq(A)

Figure 3. Algorithm with f(A) = max(A^2) and solution vector s

The first step in the algorithm of Figure 3 is to initialize the solution vector to the trivial solution s(n)={1,2,3,...n}. The algorithm then steps through a series of feasible solutions. Each time the dimension of the matrix is reduced by one the number of equivalence classes in the solution vector s is also reduced by one. The solution vector is updated by taking all variables that currently have solution value j and assigning a new solution value i. All variables that currently have a solution greater than j have their current value reduced by one. For this to work properly the subscripts {ij} must be chosen such that i<j.
There must also be a deterministic stopping criterion and a deterministic method of choosing a unique xij when more than one pair of variables has the same maximal intersection. The convention that has been adopted uses the upper triangular part of the A^2 matrix since A^2 is symmetric. If there are more than one pair of variables in {ij} choose the one with the lowest i and then the lowest j value. This uniquely determines i and j in any step of the algorithm and assures that any program implementation will go through the exact same deterministic steps. This also ensures that i<j so that the solution vector is calculated properly. The stopping criterion is that the set {ij} is empty ({ij}={ϕ}) .

7 Solution Space

The *system space* {A} is the set of all binary symmetric zero-diagonal matrices that represent any system of inequations. The solution space S of a decision function can be defined as the subset of all systems {A} where the method provides an optimal solution S={A| f(A) → s*)} where s* is an optimal solution of cardinality k*. The solution space of a decision function can be estimated using simulation as illustrated in Figure 4 for f(A) = max(A^2).

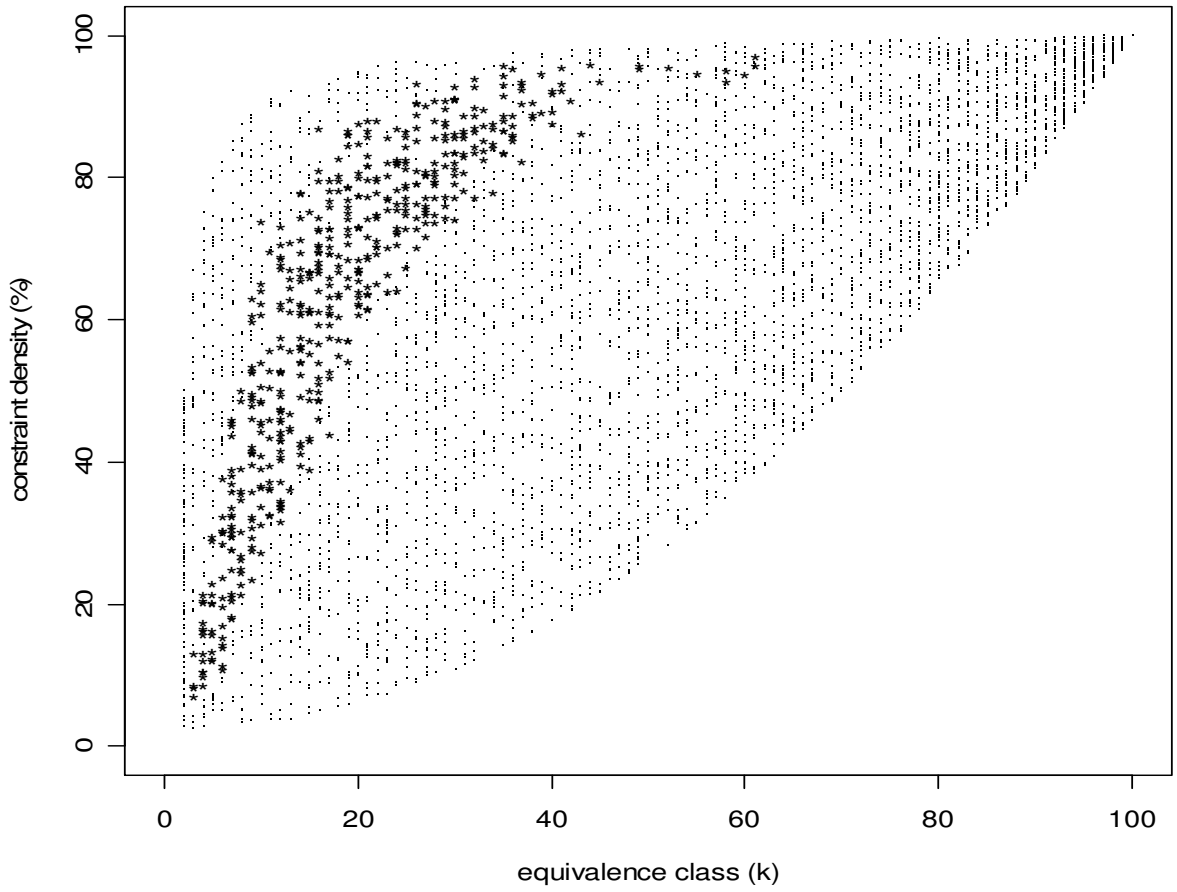

Figure 4. Solution Space S for f(A) = max(A^2)

Figure 4 was generated using the algorithm of Figure 2 for 5000 systems of dimension n=100 variables using uniform distributions of optimal solution cardinality k* and constraint density ($\sum aij$ /((n)*(n-1)). The points in Figure 4 represent optimal solutions and the asterisks represent suboptimal solutions. Of the 5000 systems of inequations represented in Figure 4 there are approximately 4500 points representing where an optimal solution was found (see Table 1) all falling into a semi-contiguous region. Systems for which an optimal solution was not found can be visualized as teeter-tottering on the edge of indecision, located in somewhere in-between the regions of absolute convergence of a number of complete k-partite "attractors". Finding the optimal solution s* for the general case where 1<=k*<=n is classified as NP-hard[Weisstein, 2005a][Horowitz, 1978]. Finding an optimal solution s* for the case of three equivalence classes (k*=3) is a special case which is classified as both NP-hard and NP-complete[Weisstein, 2005b]. A solution for either the general case 1<=k*<=n or the NP-complete special case of k*=3 would provide a solution for the entire class NP complete [Cook, 2005], [Karp, 1972].

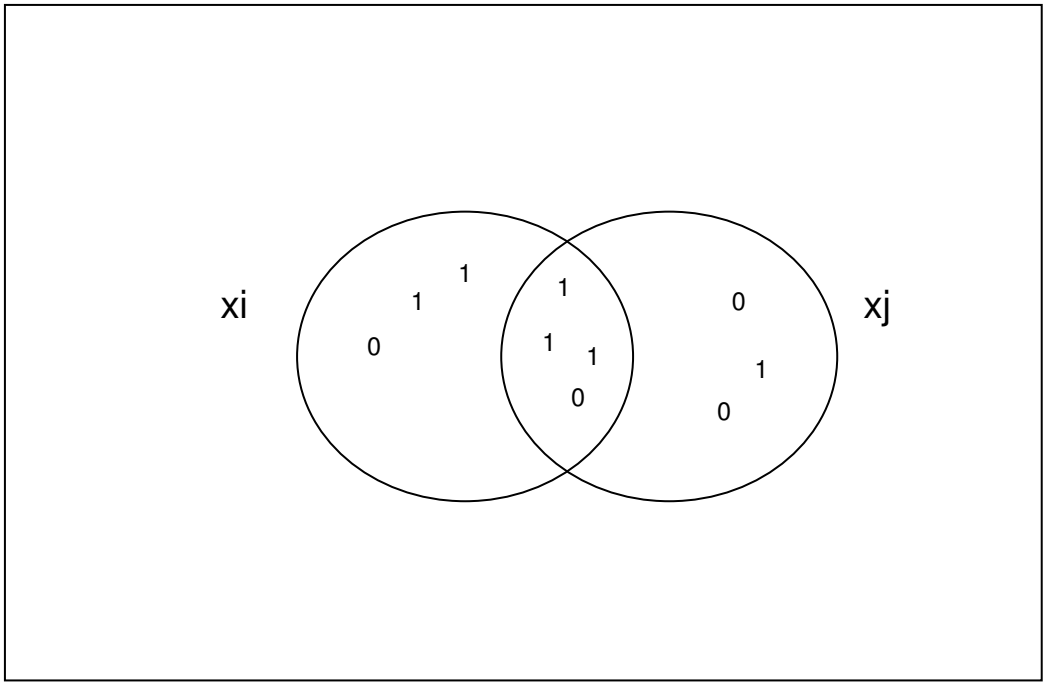

Figure 5. Venn Diagram illustrating decision functions A^2, A $\overline{A}$, and $\overline{A}^2$

The decision function f(A) = max(A^2) is only one of many that have been identified based on the powers of the matrix A and its complement $\overline{A}$. For example f(A) = max($\overline{A}^2$) is the decision function based on the number of zeros in common (i.e., the intersection of non-constraints). This decision function can be thought of as maximizing the number of variables in each equivalence class or independent set. Another decision function $f(A) = \min(\overline{A}A)$ minimizes the number of constraints added to a variable when associating it with another variable. Figure 5 illustrates these decision functions for two variables x_i=(1,1,1,0,1,0,1) and x_j=(1,1,1,0,0,1,0) whose first 4 elements agree and last 3 elements disagree. The intersection of constraints or number of ones in common for xi and xj corresponding to f(A) = max(A^2) $x_i \bullet x_j$ = 3. Similarly, for f(A) = max($\overline{A}^2$) the number of zeros in common is $\overline{x_i} \bullet \overline{x_j}$ = 1. If the constraints of x_i are added to x_j the number of constraints of x_j is increased by 2 whereas if the constraints of x_j are added to x_i the number of constraints of x_j is increased by 1 so (min($x_i \bullet \overline{x_j}, \overline{x_i} \bullet x_j$) = 1). A generating function for decision functions is given by $K^m = (A+\overline{A})^m$ where K is the complete system (k* = n). For m=2 these decision functions are related by equation (10).

$$K^2 = (A+\overline{A})^2 = A^2 + A\overline{A} + \overline{A}A + \overline{A}^2 \tag{10}$$

It is also possible to use a power series expansion to generate decision functions as in equation (11).

$$f(A) = \sum_{i=0}^{\infty} c_i A^i \tag{11}$$

Each set of c_i values generates a unique decision function. Note that both (10) and (11) include as a subset the set of decision functions f(A) = A^n .

Table 1. Estimated percent solution space S for various decision functions

decision function	n=10	20	30	50	100	200	300	500	1000
1. max(A^2)	99.5%	98.6%	96.8%	93.2%	90.0%	87.6%	86.1%	87.4%	86.8%
2. max($\overline{A}^2$)	91.3%	80.9%	73.7%	66.1%	63.4%	60.2%	58.5%	62.0%	56.7%
3. min($A\,\overline{A}$)	99.9%	98.4%	96.5%	91.3%	89.2%	88.8%	88.3%	86.6%	87.2%
4. min($A(\overline{A} * A^2)$)	99.9%	99.4%	97.2%	94.6%	92.6%	90.3%	90.0%	88.9%	88.1%
5. combined (1-4)	100%	100%	98.8%	96.4%	94.6%	92.5%	90.8%	89.8%	89.3%
6. f(A)=A	72.4%	61.2%	57.3%	56.8%	55.7%	57.9%	56.9%	56.5%	55.2%

Table 1 shows the estimated solution space of various decision functions for systems of dimension between 10 and 1000. In general the success rate drops off with increasing n but stabilizes to nearly a constant value as n increases. The success rate of f(A)=max($\overline{A}^2$) is lower than f(A)=max(A^2) while that of f(A)=min($A\,\overline{A}$) is about the same. The decision function with the best success rate f(A)=min($A(\overline{A} * A^2)$) is a variation on f(A)=min($A\,\overline{A}$) which weights the matrix $\overline{A}$ using scalar multiplication (*) with A^2. The effect of this weighting is that it discourages combining two variables if doing so would prevent one of them from combining with another variable that it is highly correlated with. The solution space for each decision function overlaps with the others but not completely so that using more than one decision function increases the overall success rate. The row of Table 1 labeled "combined" shows that calculating the first four decision functions and using the best result increases the solution space S to almost 90% for systems as large as n =1000. The last row shows the result of the ambivalent or neutral decision function f(A)=A. The resolving ability of any decision function can be measured by calculating the ratio if its success rate to that of the ambivalent decision function.

Consider any bipartite system in canonical form such as B in equation (12). Any bipartite system will have two blocks of zeros on the main diagonal so the inner product of any two variables from different equivalence classes will always be zero. If the variables are from different equivalence classes the intersection will not be equal to zero. Therefore f(A) = max(A^2) solves all bipartite and acyclic systems.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \qquad (12)$$

Observation: *the decision function f(A) = max(A^2) solves all systems of inequations with k*=2. This includes all incomplete and complete bipartite systems and all acyclic systems. In addition, it solves all complete k-partite systems and all systems close enough to complete k-partite to be within its region of convergence as in Figure 1. The solution space of f(A) = max (A^2) includes systems of dimension* $n \to \infty$.

Observation: *whenever the set of constraints of a variable xi are a subset of the constraints of another variable xj the variable xi is redundant and can be combined with xj without changing the cardinality of the optimal solution k*. This is the basis of the decision function f(A)=min($A\,\overline{A}$) which minimizes the number of new constraints added to a variable when it is combined with another variable.*

8 Equivalence Class Subset Algorithm

The equivalence class subset algorithm is based on the observation that in most cases the decisions made by a decision function f(A) to obtain a solution vector s will be correct. For example it can be estimated from Table 1 thar for n=100 the decision function $f(A)=\max(A^2)$ made a correct decision 99.5% of the time (approximately one incorrect decision out of every 200 correct decisions). This means that for a solution vector s generated by $f(A)=\max(A^2)$ any particular x_{ij} in X_{ec} is more likely to be in X_c than a given x_{ij} in X since even a suboptimal solution s found by a decision function f(A) is in some sense close to an optimal solution s*. As a consequence, a significant number of possible subsets $\{xij\} \in X_{ec}$ will be usually be chromatic (i.e., $\{xij\} \in X_c$) even when the entire set X_{ec} is not chromatic. If {xij} is chromatic it can be used to generate a new matrix A' = A(xij) having the same optimal solution cardinality as the original A matrix by associating all the pairs of variables in {xij}. The new A' = A(xij) matrix represents a node on the tree of all feasible solutions and if $\{xij\} \in X_c$ it is certain that there is at least one path below it that leads to an optimal solution. The A' = A(xij) matrix will generally be closer to complete k-partite than the matrix A and has a possibility to be in the solution space S of one of the decision functions even when the matrix A was not. This observation is the basis of a technique called the equivalence class subset algorithm which significantly improves the success rate of the decision functions listed in Table 1.

It is possible to choose a random subset {xij} of X and significantly improve the overall success rate. However it is also possible to choose a subset {xij} of X in such a way that decreases the amount of search required to find an optimal solution. To do this requires the introduction of the parameter z which is the number of block diagonal zeros in the A matrix when the rows and columns are permuted into block diagonal form (as in equation (5)). The desired expression is given by equation (13):

$$z = \sum_{i=1}^{k} k_i^2 \tag{13}$$

where k_i is the number of variables in each of the k equivalence classes. The underlying quantity of interest is the total number of pair associations X_{ec} in a solution vector s which is given by $X_{ec} = (1/2)(z - n)$ which represents half the number of off-diagonal zeros in all diagonal blocks of A. For example the solution vector s = (1,2,1,2,1) from equation(5) would have z=13 and $X_{ec} = 4$ since $k_1=3$ and $k_2=2$ as is easily verified by counting the off-diagonal zeros in equation(5). In practice equation (13) is used rather than the formula for X_{ec} since it leads to the same result. To estimate which pair associations are likely to lead to an optimal solution an nxn pair association matrix Z can be defined as shown in equation (14).

$$Z = \begin{bmatrix} 0 & \bar{z}_{12} & \bar{z}_{13} & . \\ \bar{z}_{21} & 0 & \bar{z}_{23} & . \\ \bar{z}_{31} & \bar{z}_{32} & 0 & . \\ . & . & . & . \end{bmatrix} \tag{14}$$

The element z_{ij} of the matrix Z represents an estimate of the average number of total pair associations found when the pairs i and j are in the same equivalence class. In general, the more pair associations in a solution s means the less number of equivalence classes k required. This can be seen from the formula for $z_{min} \approx k(n/k)^2 \approx n^2/k$ which is the minimum number of block diagonal zeros for solution cardinality k.

During an equivalence class subset search, many feasible solutions are found. Every time any solution s is calculated, the total number of pair associations z corresponding to s is calculated and used to update the matrix Z for the set of pair associations in s. On average pair associations with the largest values in the Z matrix are more likely to lead to optimal solutions. The resulting algorithm involves using the Z matrix to choose a maximum likelihood subset {ij} to maximize the probability of finding an optimal solution.

while(condition){
A' = A({ij})
s = ineq(A')
k = max(s)
z = z(s)
if(k==k* && z>z*) s*=s, z*=z
if(k<k*) s*=s, k*=k,z*=z}

Figure 7. Equivalence class subset algorithm

The equivalence class subset algorithm is given in Figure 7. The first step in the equivalence class subset algorithm (not shown for clarity) is to initialize s*=(1,2,…n), k*=max(s) = n, and z*=n where s*, k*, and z* represent the current best solution and n is the dimension of the system. This corresponds to the trivial solution that solves any system of inequations and also the optimal solution for k=n. The while(condition) provides a stopping criterion which limits the number of iterations to some number such as n or n^2. The first step of the algorithm is to take the system A matrix and create a new A' matrix by combining some subset of variables {ij} based on the matrix Z and the current best solution s*. The first time through the algorithm all values of zij in Z=0 and s*=(1,2,3…n) which means that {ij}={ϕ} so that A'=A. The solution vector s corresponding to A' is then calculated using the ineq(A) algorithm in Figure 2 and this solution is used to calculate the solution cardinality k=max(s). The total number of block diagonal zeros z=z(s) is then calculated according to equation (13). There are three possible outcomes: k=k*, k<k*, or k>k*. If the solution cardinality has not improved (k*=k) but the number of pair associations z has improved (z>z*) then the solution s is taken as the new best solution s*. If the solution cardinality has improved (k<k*) then the solution s is taken to be the new best solution s*. If the solution cardinality of s is greater than s* (k>k*) or if k=k* and z<z* then the solution s* is not updated. Each time an improved solution is found it increases the probability of finding an optimal solution on subsequent iterations because a subset of s'(k') is in general more likely to be chromatic than a subset of s(k) for k'<k. Before starting the next iteration the pair association matrix Z is updated using equation (14).

The selection of the subset size |{xij}| is a direct tradeoff between two opposing factors. The smaller the subset size the higher the probability that the set {xij} is jointly chromatic (each individual xij being chromatic does not guarantee that property for the set {xij}). On the other hand the larger the subset size the lower the probability that {xij} is chromatic. However the larger the subset size the higher the probability that the intermediate node on the tree of feasible solutions corresponding to the subset {xij} will be on one of the decision function paths to an optimal terminal node (i.e., A(xij) closer to complete k-partite). Experimentation has shown that subset sizes less than about 30% of the maximum value of $|X_{ec}|$ are more likely to result in an improved solution cardinality. Once the size of a subset is chosen the {xij} can be chosen using the rank ordering of $\overline{z_{ij}}$ given by the Z matrix (the higher the value the better).

The number of iterations required to find an optimal solution varies however under certain circumstances the algorithm can terminate when it is certain that an optimal solution has been found. This could happen if the cardinality of a feasible solution is found to be equal to k=2 or k=3 or the cardinality of a solution equals a lower bound on $k^*=k_{LB}$. Since the algorithm f(A) = max(A^2) always finds an optimal solution for k*=2 the algorithm can terminate if k=2. If at any point during the search the algorithm finds a solution of cardinality k=3 it can terminate immediately because it cannot be improved. If it could, it would have found a solution vector s*(k*=2) on the first pass. A lower bound $k^*=k_{LB}$ can occur in certain cases because it can be shown that for any system of inequations the optimal solution cardinality k* can never be less than the number of variables in the largest equivalence class of the complementary system $\overline{A}$. This lower bound can be obtained by using the algorithm of Figure 3 to calculate any solution vector $\bar{s}$ of the complementary system $\overline{A}$. The number of variables in the largest equivalence class of $\bar{s}$ gives k_{LB}.

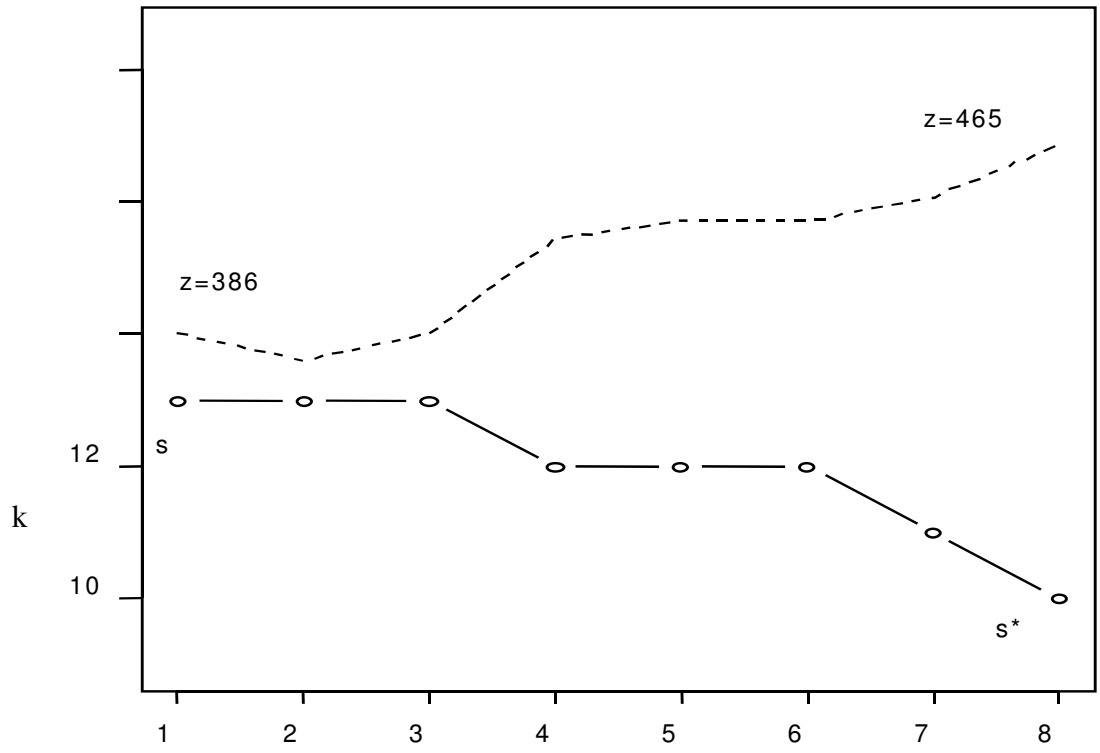

Figure 8. Equivalence Class Subset algorithm trajectory to an optimal solution s*

Figure 8 shows an example of equivalence class subset algorithm for a system of inequations in n=100 variables whose optimal solution cardinality k=10. As shown by the solid line in Figure 8 the initial solution s was of cardinality (k=13) but after only 8 iterations an optimal solution s* (k=10) was found. The dashed line shows that in this case the initial solution s had z=386 zeroes in diagonal blocks while the optimal solution s* had z=465 zeroes in its diagonal blocks. Note the method progressed through a sequence of suboptimal solutions before finding s*. Each of the suboptimal solutions contributed to updating the Z matrix which in turn generated better and better solutions.

9 Complexity and Performance

Using Figure 2 as a model it is seen that solution method for systems of inequations involves finding the union of two sets (O(n)) and removing a row and column from an nxn matrix (also O(n)). This must be done (n-k) times which gives an average of E[O(n)*(n-k)] = $O(n^2)$ operations. The calculation of the decision function f(A)=max(A^2) requires squaring a matrix which counts as $O(n^3)$ operations. This has to be repeated (n-k) times resulting in a complexity of $O(n^4)$ for the algorithm in Figure 2 (equation (15)).

$$\sum_{i=0}^{n}(n-i)^3 = \frac{(n+1)^2(n+2)^2}{4} = O(n^4) \tag{15}$$

Use of the equivalence class subset algorithm can increase the complexity beyond $O(n^4)$. For example, if the number of iterations is O(n) then the resulting complexity would be $O(n)*O(n^4) = O(n^5)$.

The result of tests on the equivalence class subset algorithm for systems of dimension n=100 show a 99.8% success rate for 5000 iterations with a 99.3% success rate after only 500 iterations. A geometric model can be used to estimate the probability that an optimal solution has been found for the equivalence class subset algorithm after a certain number (m) of iterations. The probability of finding an optimal solution after m iterations would then be given to a first order approximation as the solution to $P[s^*|m] = .993 = 1-q^m$ with m=500. This gives a value of q=.99 and $P[s^*|m] \cong 1$ for m=10,000. For n=100 a value of m=10,000 is $O(n^2)$ so that it is reasonably certain that an optimal solution s* has been found in a complexity of $O(n^4)*O(n^2)=O(n^6)$. Unfortunately the geometric model is only an approximation and it is not clear how accurate the estimate of success rate is for very large numbers of iterations.

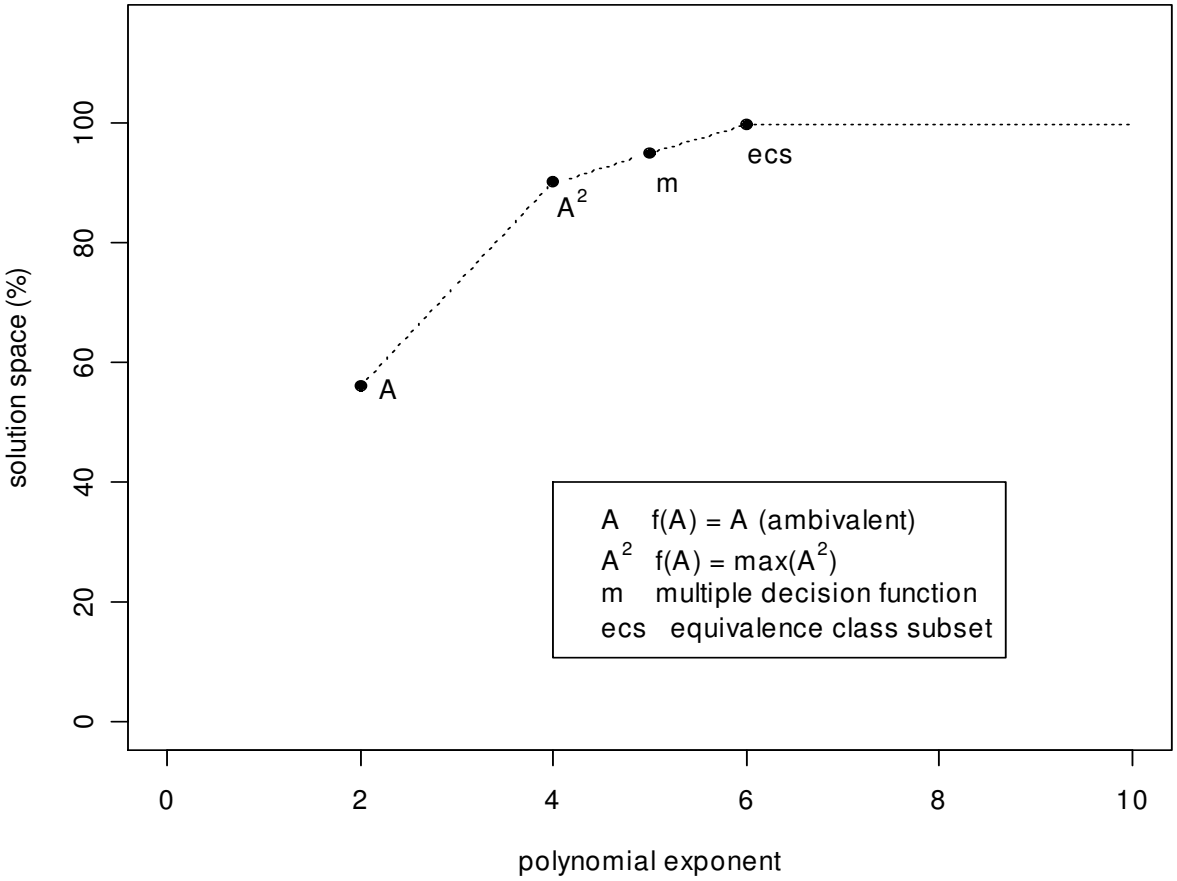

Figure 9. Algorithm success rate vs. number of computations for n=100

Figure 9 shows the success rate for various methods compared to the complexity of computation for systems of dimension n=100. The ambivalent decision function f(A)=A has a success rate of about 56% for $O(n^2)$ operations. The decision function $f(A)=\max(A^2)$ is of complexity $O(n^4)$ and has about a 90% success rate. The multiple decision function (m) represents the combined result (94.6%) in Table 1 for more than one decision function which was conservatively assigned a complexity of $O(n^5)$. The equivalence class subset algorithm (ecs) raises the success rate to approximately 99.8% for complexity $O(n^6)$. Preliminary calulations with systems of dimension n=200 show a drop off in success rate to around 99.5%. It is not known how high the success rate can be raised using the equivalence class subset method or any other polynomial time method. What is known is that the success rate vs. number of iteration curve is a very long-tailed distribution. The shape of the curve in Figure 9 suggests that it may be possible for some algorithm to asymptotically approach a success rate of 100% in polynomial time complexity.

10 Summary and Conclusion

Systems of inequations are equivalent to the compatibility problem[Clay,2005] which is one of the best known and classic examples of an NP-hard/complete problem. The same method used for solving systems of equations, substitution and elimination of variables, can be used to solve systems of inequations. A decision function is used to determine which variables to combine. The algorithm can be implemented with set-theoretical operations such as union and intersection or with modulo and arithmetic operators used for solving systems of equations. The decision function $f(A) = \max(A^2)$ can be shown to solve a significant subset of all systems of inequations for arbitrary solution cardinality k and dimension $n \rightarrow \infty$. The equivalence class subset algorithm is an extension of the general solution method based on the fundamental observation that in general optimal solutions have more block-diagonal zeroes (z) than suboptimal solutions. This leads to the idea of defining a probability matrix Z which contains for each pair association x_{ij} the average number of zeros for all solution vectors in which x_{ij} has appeared. This probability matrix Z is then used to choose a subset {xij} with the maximum likelihood of generating an optimal solution vector. The equivalence class subset algorithm calculates an initial solution vector s and then moves back up the tree of feasible solutions to search for an improved solution based on a subset of the decisions that were made to reach the initial solution vector. Each time the algorithm finds an improved solution it further increases the probability of finding an optimal solution since any subset of an improved solution is more likely to contain only correct decisions. In addition each new matrix A' generated has a chance of being in the solution space S of at least one decision function even when the matrix A was not. The principal parameters affecting the performance of the equivalence class subset algorithm are the number of decision functions, the type of decision functions, the number of iterations and the method of choosing a subset. The solution method is very general in that it can be applied to any system of inequations of any solution cardinality k and system dimension n. The success rate of the equivalence class subset algorithm (99.8% for n=100) suggests that there may exist some algorithm which asymptotically approaches 100% success rate in polynomial time. The solution method for systems of inequations can also be used to solve other NP-complete problems[Cook, 2005], [Karp,1972]. This is possible because any NP-complete problem can be converted to an equivalent system of inequations in polynomial time. The equivalence class subset algorithm can be used to find a solution which can then be converted back in polynomial time to a solution for the original problem.

11 References

[1] Clay Mathematics Institute, Millennium Prize Problems, "P vs. NP", http://www.claymath.org/millenium/P_vs_NP, 2005.

[2] S. Cook, "The P vs. NP Problem", Clay Mathematics Institute, http://www.claymath.org/millennium/P_vs_NP/Official_Problem_Description.pdf, 2005.

[3] R. M. Karp, "Reducibility Among Combinatorial Problems", In Complexity of Computer Computation, pages 85-104. Plenum Press, New York, 1972.

[4] Weisstein, E. W., "NP-Hard Problem." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/NP-HardProblem.html

[5] Weisstein, E. W., "NP-Complete Problem." From *MathWorld*--A Wolfram Web Resource. http://mathworld.wolfram.com/NP-CompleteProblem.html

[6] E. Horowitz, S. Sahni, "Fundamentals of Computer Algorithms", Computer Science Press, Maryland, 1978.

12 Biographical Information

Dr. Jeffrey L. Duffany is a professor of Electrical and Computer Engineering at the Universidad del Turabo. Dr. Duffany's research interests span across a number of areas of communication and information science including algorithms, chaos theory, network security and wireless communication.