Equivalence Class Subset Algorithm

Jeffrey L. Duffany, Ph.D.

***Abstract*—**The equivalence class subset algorithm is a powerful tool for solving a wide variety of constraint satisfaction problems and is based on the use of a decision function which has a very high but not perfect accuracy. Perfect accuracy is not required in the decision function as even a suboptimal solution contains valuable information that can be used to help find an optimal solution. In the hardest problems, the decision function can break down leading to a suboptimal solution where there are more equivalence classes than are necessary and which can be viewed as a mixture of good decision and bad decisions. By choosing a subset of the decisions made in reaching a suboptimal solution an iterative technique can lead to an optimal solution, using series of steadily improved suboptimal solutions. The goal is to reach an optimal solution as quickly as possible. Various techniques for choosing the decision subset are evaluated.

***Keywords*—** np-complete, complexity, algorithm.

I. Introduction

Consider the well-known compatibility problem, also known as a system of inequations[4,5,10,11] which is one of the best known examples of an NP-hard problem[2,6,8] which include the set of NP-complete problems as a special case[7]. The compatibility problem can be stated as follows. Suppose there are n objects each of which are incompatible with some subset of the other n-1 objects. The problem is to partition all n objects into a set of k equivalence classes such that no object is incompatible with any other object in its equivalence class and where k is minimum.

The problem can be stated in terms of an adjacency matrix (A) of ones and zeros which summarizes the compatibility of each object (variable xi) with every other object (variable xj). A one in the (i,j) element of the A matrix indicates incompatibility between variables xi and xj while a zero represents compatibility. A solution vector (s) is a mapping of each variable xi into an integer si such that 1<= si <= k while ensuring that si ≠ sj when A[i,j] = 1. If an optimal solution vector s* is permuted such that all equivalence classes are grouped together and the corresponding A matrix is permuted accordingly, it is seen that the A matrix takes on a block diagonal form. A zero inside the block diagonal represents a good decision which leads towards an optimal solution. A zero outside the block diagonal or any zero of a suboptimal solution may be good or bad and may or may not lead to an optimal solution. A typical problem of n=100 variables has n^2 = 10,000 elements each of which is a zero or a one. For example a system may have 5000 zeros of which 3000 are inside the block diagonal (z_{in}) and 2000 which are outside (z_{out}). Each zero represents a possible decision and at each step of the solution only one of these can be chosen. Clearly there can be a large number of choices and some kind of decision function is required to proceed towards a solution. A pair association {ij} occurs when a pair of variables xi and xj are combined into the same equivalence class. Generally speaking the lower the solution cardinality (k) the greater the number of zeros in the blocks of the block diagonal form.

Consider a decision function f(A) =max(A^2) which is the maximum value of the square of the A matrix over all (i,j) elements where A(i,j)=0 (all pairs of variables that can be combined to form the basis of an equivalence class). This is also known as the correlation function which represents for each (i,j) the number of constraints that match between the variables xi and xj. Using the ineq algorithm (described in section IV) it is possible to find an optimal solution vector s* and permute the A matrix into block diagonal form. Now compute A^2 and take only the values corresponding to A(i,j)=0 and place all of these values into two groups: z_{in} for those inside the block diagonal and z_{out} for those outside and plot them as a histogram. The result for four different A matrices of four different ones densities is shown in Figures(1)-(4). The solid line in each of the figures represents the distribution of correlation values for pairs of variables xi and xj inside the block diagonal z_{in} and the dashed line is for those pair associations xi and xj outside the block diagonal z_{out}. The area underneath each curve represents the number of zeros either inside or outside of the diagonal blocks.

Figure 1 shows the result of squaring an A matrix of very high ones density while Figure 2 shows what happens when that ones density is reduced by about half. In Figure 1 all of the zeros are in the block diagonal and none outside which is known as a complete k-partite system. It can be seen in this case that there are three equivalence classes of different sizes resulting in three separate spectral peaks. The decision function f(A) =max(A^2) tells us to choose the pair association corresponding to the rightmost tail of the distribution which has a value of about f(A) = max(A^2) = 42.

J. L. Duffany is with the Electrical and Computer Engineering Department, Universidad del Turabo, Gurabo, PR 00778 USA, (e-mail: jduffany@suagm.edu).

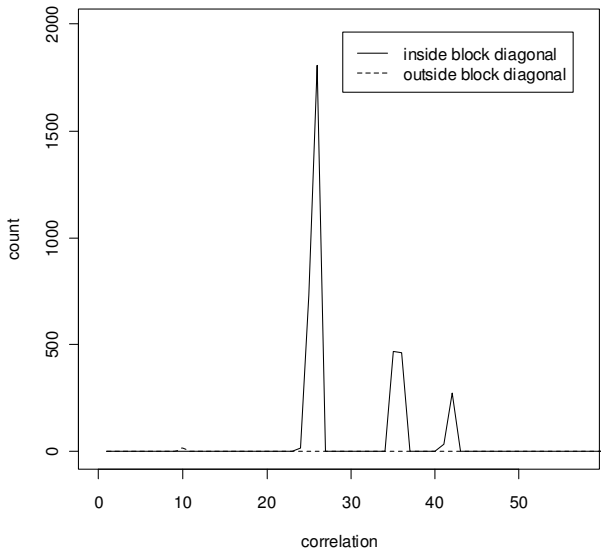

Fig. 1 Matrix A with very high constraint density

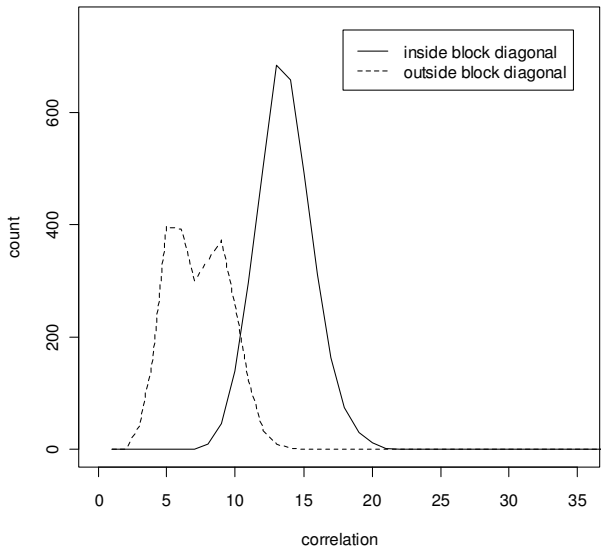

Fig. 2 Matrix A with average constraint density

Because this is a complete k-partite system choosing any valid pair association will lead to an optimal solution so any decision function could be used in this case. This is a special case where any decision is a good decision and there are no bad decisions. There are other decision functions such as a power series expansion of the A matrix $f(A) = \max(\sum c_i A^i)$ which is explored in more depth in [5].

Figure 2 shows the result for an A matrix with average ones density where the zeros outside of the main block diagonal increase significantly. However the decision function $f(A) = \max(A^2)$ can clearly still differentiate between good decisions which lead to an optimal solution (under the solid curve) and other decisions or pair associations which may or may not lead to an optimal solution (under the dashed curve). The decision function will continue to work as long as the tail of the solid curve extends further to the right than the dashed curve since it only needs to make one correct decision at a time. The decision function $f(A) = \max(A^2)$ is not perfect it has been estimated to be correct about 99.5% of the time over the ensemble of systems of dimension n=100. Usually an optimal solution will be found for any system similar to the one in Figure 2 because as the solution proceeds the system gets closer to a complete k-partite system (Figure 1).

Figure 3 illustrates what happens when the number of zeros z_{out} continues to increase to the point where it equals or exceeds z_{in}. It can be seen in Figure 3 the two distributions do overlap however the right hand tail of the solid line distribution extends significantly higher than the dashed distribution so the decision function still works. Figure 4 shows the extreme case where there are far more zeros outside the block diagonal z_{out} than inside z_{in}.

Note in Figure 4 that the two distributions are almost perfectly overlapping causing the decision function to break down and lose its ability to discriminate. This also corresponds to a very low ones density which is known to be the where the most difficult problems are[1,3]. This provides clear insight as to why the problems are so hard at this constraint density.

It is important to note that all problems start near to one of the situations (Figures (1)-(4)) and then progresses towards a complete k-partite as in Figure 1. That explains why most of these problems are easily solved except for low density problems such as in Figure 4. Each system progresses along a trajectory towards a solution and will always pass through and end up as a complete k-partite system[3] as in Figure 1.

II. Equivalence Class Subset Algorithm

Consider the case of a typical np-complete problem which must determine if a given set of variables can be represented as three equivalence classes (k*=3) or not. If a wrong decision is made at any point during the solution it will find a suboptimal solution of more than k*=3 equivalence classes. In the more general case of the np-hard compatibility problem the optimal solution is k* equivalence classes and the algorithm finds a suboptimal solution of cardinality k >k*. One technique to improve the success rate is to assume that the suboptimal solution is somehow close to an optimal solution and that most of the decisions made by the decision function were correct. The method chooses some subset of the decisions that were made in reaching the suboptimal solution and combines those variables creating a problem that does fall into the optimal solution space of the decision function.

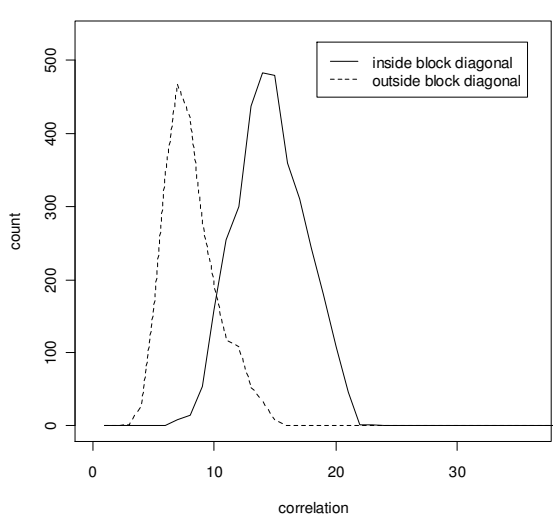

Fig. 3 Matrix A with low constraint density

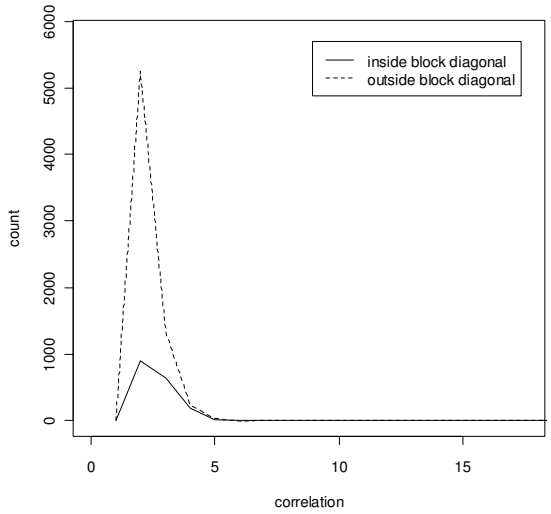

Fig. 4 Matrix A with very low constraint density

The central idea is that somewhere near every system A that cannot be solved directly in one iteration is another equivalent system A' with the same optimal solution as system A which can be solved in one iteration. Finding this system A' can be viewed as a search of the very top of the decision tree. The equivalence class subset algorithm can be stated as in Figure 5:

(1) find any solution s to system A

(2) choose a subset of variable associations from s

(3) combine these variables to create the A' matrix

(4) find a solution s' to system A'

Figure 5. The Equivalence Class Subset Algorithm

The equivalence class subset algorithm is based on the observation that any solution vector s will partition the zeros of the A matrix into two groups: those inside the block diagonal z_{in} and those outside the block diagonal z_{out}. Combining a small subset of pair associations corresponding to zeros inside the block diagonal may lead to an optimal solution if all of the decisions they represent were good. The exact size of the subset can be referred to as the search depth. Easier problems are solved by combining variables in the same equivalence class while difficult problems can require combining variables in different equivalence classes.

The equivalence class subset algorithm works by stepping through a sequence of suboptimal solutions towards an optimal solution. Each time a solution s' is found the number of zeros on the block diagonal (z_{in}) is calculated. If this is more than in the previous solution use s' to take the next subset otherwise continue using s. If you take two suboptimal solutions it is likely that that one has more zeros on the main diagonal than the other. The one with more zeros is likely to be closer to an optimal solution. Each time you find a system where the number of zeros on the block diagonal is greater than any yet calculated, choose that as the new system s'.

Figure 6 shows a typical trajectory of the equivalence class subset algorithm from suboptimal to optimal solution for a system of n=100 variables and optimal solution cardinality k*=13. The lower line represents the sequence of k values starting at k=15 and ending an optimal solution k*=13. The dashed line represents the total number of pair associations z_{in} which it is desired to maximize (for simplicity the actual numbers are not shown for the pair associations). The method required choosing about 100 subsets from suboptimal solution vectors before an optimal solution was found. Usually an optimal solution is found in fewer attempts and occasionally it can take several hundred attempts[3] for systems of dimension n=100. The optimal size for the subset is a parameter of great interest and has been studied in [3]. The larger the size of the subset, the higher the probability it is to contain an incorrect decision. The smaller the size of the subset, the less likely it is to contain an incorrect decision however the less likely it is to move the system over to one that is in the solution space of the decision function. For systems of dimension n=100 it has been seen a range of subsets from as few as 1 to as many as 9 falling off rapidly in what appears to be a geometric type of distribution[3].

The assumptions behind the equivalence class subset algorithm that a suboptimal solution is somehow close to an optimal solution may not hold for a small set of the most difficult problems which are typically of low ones density as seen in Figure 4. This is due to a type of avalanche effect whereby mistakes made by a decision function early on can lead to suboptimal solutions that are far away from an optimal solution. In this situation it can be hopeless to combine any subset of variables as the probability of including a bad decision is very high. In this situation it would be better to

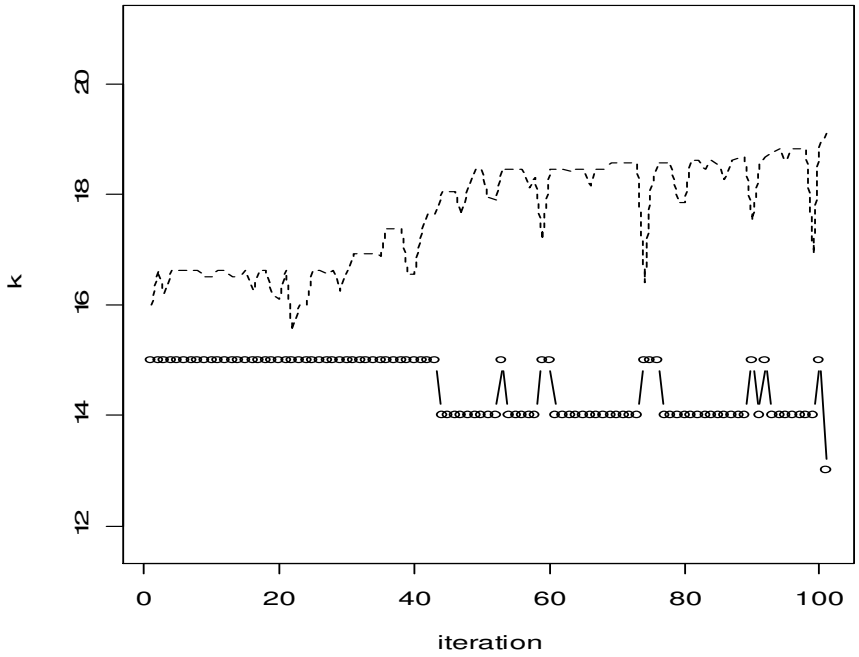

Figure 6. Typical Trajectory of the Equivalence Class Subset Algorithm

assume that the suboptimal solution is far from an optimal solution and do the opposite. Instead of combining variables from the same equivalence class the method should split variables under the assumption that a significant proportion of the decisions made were probably wrong. For example, variables combined in the smallest equivalence class can act like a set of misfits that should be split up rather than combined in the equivalence class subset algorithm.

That raises the problem of how to separate the split pair associations so that they don't recombine. This can be done quite effectively by creating a list of all pair associations that were previously combined and the resulting value of z_{in} that resulted from doing this. When a pair association needs to be split it is looked up on the list and combined with a variable it is most compatible with. Use of this technique has extended the capability of the equivalence class subset algorithm to solve very difficult problems. Since the difficulty of a given problem is not known a priori, techniques of both splitting and combining can be used in an alternate fashion.

III. Problem Complexity

Considerable discussion has been put forth already on the ones density of the A matrix and the number of zeros inside and outside of the block diagonal. From a purely probabilistic viewpoint the odds of making a good decision depends on the underlying ratio of good to bad which is related to the number of zeros inside and outside of the block diagonal. The decision function works most of the time except in low ones density problems as in Figure 4. By contrast, in a complete k-partite system (the opposite extreme) there are only good decisions. For any given system and optimal solution vector s* there will be a zozi ratio given by equation (1):

$$\text{zozi} = z_{out}/z_{in} \quad (1)$$

In general it makes intuitive sense that the higher the zozi ratio the more difficult the problem. Recall for example that the easiest problems all have a zozi ratio of zozi=0. Difficult can be defined as the average time complexity of solving a problem[3] or as the difficulty of guessing at random and getting the correct answer. Figure 7 shows the result from a sample of 200 systems all in the most difficult region of low density but with different zozi ratios. It was observed that all of the most difficult problems had a zozi ratio of greater than about 1.8. The zozi ratio for k*=3 can vary between about .0006 and 2. The difficulty tapered off as the zozi ratio decreased from 1.8 to 1.5 and for zozi ratios less than about 1.5 all were solved in one iteration. It is premature to reach a general conclusion from a small sample of 200 however there is a clear indication that most if not all of the difficult problems are concentrated in the region with the highest ratio of zeros outside z_{out} to zeros inside the block diagonal z_{in}. This leads to the question of how it is possible to construct problems which have the highest ratio of zeros outside to zeros inside the block diagonal. Simply decreasing the ones

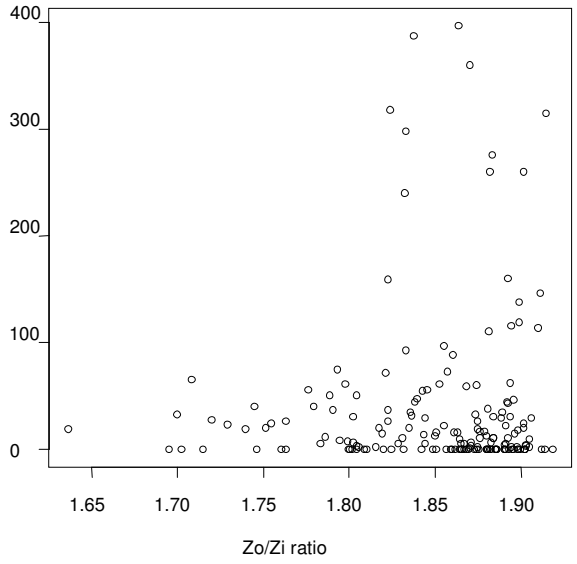

Figure 7. Problem Complexity and Number of Zeros z_{in} and z_{out}

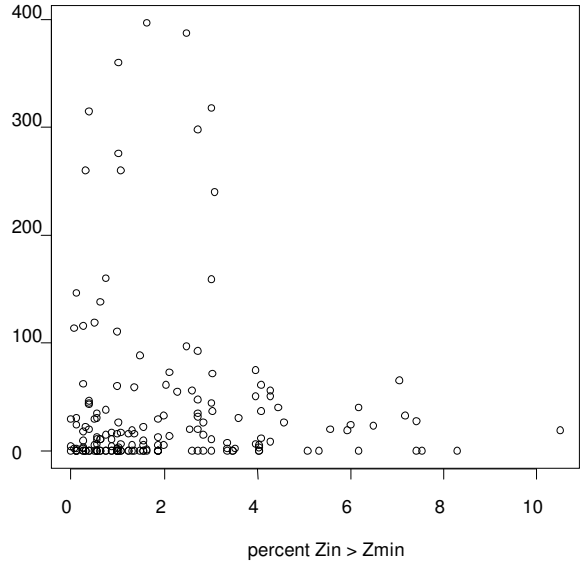

Figure 8. Problem Complexity and Number of Zeros $z_{in} > z_{min}$

density does not necessarily accomplish the desired result as it can result in the trivial problem where the ones density is zero. Referring to equation (1) it can be seen that to maximize the zozi ratio it would be necessary to consider both maximizing the numerator and minimizing the denominator of the fraction. The formula for calculating the minimum number zeros in the block diagonal is given by equation (2):

$$z_{min} = k*(n/k)^2 = n^2/k \quad (2)$$

which gives approximately the minimum number of block diagonal zeros for solution cardinality k (exactly if n is divisible by k). This tells us that the minimum number of zeros on the block diagonal occurs when the size of all the equivalence classes are the same (n/k). For n=100 and k=3 this gives about 3334 as the minimum and it is for these problems that would be expected to be the most difficult. To test this hypothesis a sample of 200 systems were chosen and solved. For each system the ratio z_{in}/z_{min} was plotted against complexity as shown in Figure 8. All of the most difficult problems were seen to fall very close (within 4%) of z_{min}.

IV. The ineq Algorithm

The decision function f(A) =max(A^2) and the equivalence class subset algorithm were discussed in some detail. However this does not give the details of how to use a decision function by itself to generate a solution vector s as required in step (1) and step (4) of the equivalence class subset algorithm as given in Figure 5. The method used to generate solutions is called the ineq algorithm (see Figure 9).

```
ineq(A)
ij<-max(A^2)
xi=xi|xj
s[j]=s[i]
A=A[-j,-j]
ineq(A)
```

Figure 9. The ineq algorithm

The ineq algorithm is so named because it can solve systems of inequations[4][10][11]. The algorithm starts with a solution vector s which has an initial value of s = (1,2,3,...n). The algorithm squares the adjacency matrix A and finds the maximum value of A^2 [i,j] for pairs of variables that can be combined (i.e., A[i,j]=0). Similar to gaussian elimination it then combines variables xi and xj by taking the constraints that are in xj but not in xi and adding them to xi. It then updates the solution vector s[j]=s[i] and eliminates variable xj as in Gaussian elimination. The matrix A is reduced by one in dimension each time a variable is eliminated. Note that the algorithm ineq is recursive, calling itself in the last step of the algorithm and stopping when there are no longer any variables to combine. The difference between ineq and Gaussian elimination is that ineq uses logical OR instead of addition and uses a decision function f(A)=max(A^2) to determine (i,j). Since squaring a matrix is complexity $O(n^3)$ it would appear that ineq would be $O(n^4)$ however it is shown in [12] that ineq is $O(n^3)$. This has to be repeated for O(n) repetitions giving $O(n^3)*O(n) = O(n^4)$ for the equivalence class subset algorithm.

V. Summary and Conclusions

A method called the equivalence class subset algorithm is described which can solve a wide variety of constraint satisfaction problems even in the most difficult ones[1]. It is based on the idea of the use of an imperfect decision function and by successively improving suboptimal solutions until an optimal solution is found. It was found that two different techniques were required depending on how far the suboptimal solution is from an optimal solution. If the suboptimal solution was close to an optimal solution then most of the decisions it made were correct so that choosing a subset of pair associations from even a suboptimal solution s will in many cases only contain pair associations which lead toward an optimal solution. When the subset of pair associations {ij} are associated by combining the indicated variables the matrix A is modified to form a new matrix A'. If no incorrect decisions were included in {ij} then the A' matrix will have the same optimal solution cardinality as the matrix A. If in addition the matrix A' falls into the solution space of the decision function f(A) being used then an optimal solution will also be found. In the case where the suboptimal solution is very far from an optimal solution the opposite approach of splitting up pair associations was seen to be far more successful. For any given problem it is not known whether a solution is optimal so in those cases it appears best to alternate between the two techniques. Each time a new solution is generated the number of block diagonal zeros z_{in} is calculated. If the new solution vector has a higher z_{in} value than previous solutions it is kept as the new basis for choosing subsets to combine or split. If not it is discarded. This method has been used on thousands of problems in the most difficult problem regions and has found an optimal solution in every case within at most a few hundred iterations. This leads to a conservative estimate for a lower bound of success rate of over 99.9% for systems of n=100 variables[3]. This investigation also further explored the idea of characterizing the most difficult problems[1,3]. Any insight into the nature of the most difficult problems can lead to either new or improved methods for solving these types of problems. It was found that one of the main source of problem difficulty is a result of the ratio of the number of zeros inside the block diagonal z_{in} to those outside the block diagonal z_{out} which can also be called the zozi ratio. It was further determined that the problems with the smallest number of zeros in the block diagonal would also be among the most difficult and this was corroborated by experimental results. Future research in this area is planned to include extending the dimension of the problems to be solved into the range of several thousand variables. Another goal is to test this algorithm more thoroughly by applying it to known problems for example the standard set of problems at the Carnegie Mellon University mathematics website[13]. It is also desired to create a package for the R language[14] for the ineq and ecsa algorithms so any results can be independently verified.

References

[1] S. Cook and D. Mitchell. (1997). "Finding Hard Instances of the Satisfiability Problem: A Survey", Satisfiability Problems: Theory and Applications. American Mathematical Society, Providence, Rhode Island.

[2] C.H. Papadimitrou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", Dover, ISBM 0-486-40258-4, pp. 344.

[3] Duffany, J.L., "Statistical Characterization of NP-Complete Problems", Foundations of Computer Science Conference, World Computer Congress, Las Vegas, Nevada, July 14-17, 2008.

[4] Duffany, J.L. "Systems of Inequations", 4th LACCEI Conference, Mayaguez, PR, June 21-23, 2006.

[5] Duffany, J.L. "Generalized Decision Function and Gradient Search Technique for NP-Complete Problems", XXXII CLEI Conference, Santiago Chile, August 20-23, 2006.

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

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

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

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

[10] Weisstein, E. W., "Inequation." From MathWorld--A Wolfram Web Resource. http:// mathworld.wolfram.com /Inequation.html.

[11] Wikipedia – Inequation page: http://en.wikipedia.org /wiki/Inequation.

[12] Duffany, J.L., "Optimal Solution of Constraint Satisfaction Problems", International Conference on Applied Computer Science, Sharm el Sheik, Egypt, January, 2009.

[13] http://mat.gsia.cmu.edu/COLOR/instances.html

[14] http://www.r-project.org

Jeffrey L. Duffany, Ph.D (M'77) became a Member (M) of **IEEE** in 1977. Dr. Duffany was born in Waterbury, CT and received the BSEE degree from the University of Connecticut in 1977. Dr. Duffany received the MSEE degree from Columbia University in 1979 and a dual Ph.D. in Computer and Information Engineering from Stevens Institute of Technology in 1996.

He joined the Bell Laboratories as a member of technical staff in 1977 where he worked in research and development publishing over 100 internal technical reports and receiving two US patents. He also spent a year working at Lucent Espana in Madrid, Spain. He joined the Universidad del Turabo in Gurabo PR in 2003. In 2005 he was a visiting scientist at Sandia National Laboratories Center for Cyber Defense in Albuquerque, NM. In 2006 he was a visiting faculty researcher at the University of Southern California Information Sciences Institute in Marina del Rey California. In 2008 he was named Office of Naval Research Faculty fellow and worked at the SPAWAR (Space and Naval Warfare Systems Center) in San Diego California. Currently he teaching graduate and undergraduate classes in computer science, electrical engineering and computer engineering.

Currently Dr. Duffany is pursuing research interests in the areas of computer algorithms, artificial intelligence and network/computer security. Dr. Duffany is a member of the IEEE and the ACM.