Finding the Maximum Weight Feasible Subsystem of an Infeasible System of Linear Inequalities

Mark Parker[1]
Mathematics Department
University of Colorado at Denver
Post Office Box 173364
Denver CO 80217-3364

Jennifer Ryan
US West Technologies
4001 Discovery Drive
Boulder CO 80303

Abstract

Given an inconsistent set of inequalities $Ax \leq b$, the *irreducible inconsistent subsystems* (IISs) designate subsets of the inequalities such that at least one member of each subset must be deleted in order to achieve a feasible system. By solving a set covering problem over the IISs, one can determine the minimum weight set of inequalities that must be deleted in order to achieve feasibility. Since the number of IISs is generally exponential in the size of the original subsystem, we generate the IISs only when they are violated by a trial solution. Computational results on the NETLIB infeasible LP library are given.

[1] This research was partially supported by Air Force Office of Scientific Research and Office of Naval Research Contract #F49620-92-J-0248-DEF.

1 Introduction

Let A be an $m \times n$ real matrix and let $\mathbf{b}$ be a real m–vector. Suppose that the system $S \equiv A\mathbf{x} \leq \mathbf{b}$ is inconsistent. A subsystem $A'\mathbf{x} \leq \mathbf{b}'$ is an *irreducible inconsistent subsystem* (IIS) if it is inconsistent and if it has no inconsistent proper subsystem. In general, an inconsistent system will have many overlapping IISs; Chakravarti [2] proved that there can exist an exponential number of IISs in an infeasible system. In order to achieve a consistent system, at least one inequality must be dropped from every IIS.

Suppose a modeler has an infeasible linear programming problem. Many researchers have proposed identifying an IIS and using this to assist the modeler in debugging her formulation. Greenberg ([10], [11], [12], [13], [14]), Greenberg and Murphy ([15]), and Chinneck ([5], [3], [4]) have been instrumental in promoting IIS debugging concepts and theory. However an IIS can contain as many as $n + 1$ inequalities, and this information is potentially useless. An interesting problem is to identify the maximum cardinality feasible subsystem. A more practical version allows the modeler to weight the constraints according to importance or flexibility. Then the maximum weight feasible subsystem is sought. Equivalently, we want to identify the *minimum weight set of inequalities that covers all IISs*, which we shall call the *minimum weight IIS cover*. If we knew what the IISs of a system S were, we could formulate the following set covering problem, where c_i is the weight on the ith constraint.

$$\begin{array}{ll} \min & \sum_{i=1}^{m} c_i z_i \\ \text{subject to} & \\ & \sum_{i \in I} z_i \quad \geq 1 \quad \text{for all IISs } I. \\ & z_i \text{ binary} \end{array}$$

Unfortunately, as discussed above, the number of IISs may be exponential in the size of the original problem, so we do not want to write down the whole problem at once. Instead, we will generate constraints dynamically for the problem and solve it iteratively as outlined below:

1. Identify an initial set of IISs J. (J may be empty.)

2. Solve the covering problem

$$\begin{array}{ll} \min & \sum_{i=1}^{m} c_i z_i \\ \text{subject to} & \\ & \sum_{i \in I} z_i \quad \geq 1 \quad \text{for all IISs } I \in J. \\ & z_i \text{ binary} \end{array}$$

Let T index the elements in the optimal cover.

3. Look for an IIS of P not covered by T. If there is none, STOP, T is an optimal cover of *all* IISs. Otherwise, add the new IIS to J and go to 2.

2 Background on Identifying IISs

IISs, also called *minimal infeasible systems* and *minimal unsolvable systems*, were first introduced in the context of linear inequality theory in the early part of this century. In his doctoral dissertation [16], Motzkin derives several results pertaining to systems of inequalities, the most interesting in this context is a necessary condition for an inconsistent system to be an IIS.

Theorem 1 ([16]): *The coefficient matrix A of a minimal infeasible system $\{A\mathbf{x} \leq \mathbf{b}\}$, where $A \in R^{m \times n}$, $\mathbf{x} \in R^n$, and $\mathbf{b} \in R^m$, has rank $m-1$.*

Motzkin proves that if the rank of the coefficient matrix of an infeasible system is one less than the number of constraints in the system, then every proper subsystem of the infeasible system is feasible, and thus the system is an IIS.

Fan [7] consolidated many of the known results for systems of inequalities. Additionally, he presents the following strengthening of Motzkin's characterization of an IIS to provide necessary and sufficient conditions for a system to be an IIS:

Theorem 2 ([7]): *The system $\{A\mathbf{x} \leq \mathbf{b}\}$, where $A \in R^{m \times n}$, $\mathbf{x} \in R^n$, and $\mathbf{b} \in R^m$, is an IIS if, and only if, the following conditions hold.*

1. *There exists exactly $m-1$ linearly independent rows.*
2. *There exists $\mathbf{y} \in R^m$ such that $\mathbf{y}^T A = 0, \quad \mathbf{y}^T \mathbf{b} < 0$, and $\mathbf{y} > 0$.*

Fan extends Motzkin's results by noting that necessary and sufficient conditions for a system to be infeasible is the existence of a solution to the alternative system (Farkas' Theorem).

Van Loon [22] was the first to identify IISs with linear programming infeasibility analysis. He interprets the result of Fan in light of the simplex method and describes a method for identifying all IISs by enumerating all bases of the system $\{(\mathbf{x}, \mathbf{s}) \in Q^{n+m} \mid A\mathbf{x} + I\mathbf{s} = \mathbf{b}\}$.

Recently, Chinneck (e.g. [5], [3], [4]) has developed a set of algorithms to identify a small cardinality IIS. These algorithms are based upon using elastic programming

on subsets of constraints to isolate infeasibility and using a deletion filter to isolate an I IS. These algorithms are now implemented in the commercial LP solvers MINOS, CPLEX, and LINDO.

To identify I ISs, we will make use of the following theorem from Gleeson and Ryan [9]. This theorem provides insight into a geometric approach to I IS identification that is based upon Farkas' Theorem of the Alternative and basic polyhedral theory.

Theorem 3 ([9]): *Let $A\mathbf{x} \leq \mathbf{b}$ denote an inconsistent set of inequalities. Then the I ISs are in 1-1 correspondence with the extreme points of the polyhedron $P = \{\mathbf{y} \in \Re^m \mid \mathbf{y}^T A = 0,\ \mathbf{y}^T\mathbf{b} \leq -1,\ \mathbf{y} \geq 0\}$. In particular, the nonzero components of any extreme point of P index an I IS.*

Recall a variant of Farkas' Theorem of the Alternative:

Theorem 4: *Exactly one of the following holds:*

1. *$A\mathbf{x} \leq \mathbf{b}$ is consistent.*
2. *There exists $\mathbf{y} \in R^m$ such that $\mathbf{y}^T A = 0$, $\mathbf{y}^T\mathbf{b} < 0$, and $\mathbf{y} \geq 0$.*

We note that the polyhedron P of Theorem 3 is a bounded form of the cone of the alternative system (2) from Theorem 4. The following form of Theorem 3 is also implied in [9]:

Theorem 3 [cone]: *Let $A\mathbf{x} \leq \mathbf{b}$ denote an inconsistent set of inequalities. Then the I ISs are in 1-1 correspondence with the extreme rays of the cone $P' = \{\mathbf{y} \in \Re^m \mid \mathbf{y}^T A = 0,\ \mathbf{y}^T\mathbf{b} < 0,\ \mathbf{y} \geq 0\}$. In particular, the nonzero components of any extreme ray of P' index an I IS.*

We shall explore further the significance of the conical version of Theorem 3 later. For now, we examine the utility of the bounded version of Theorem 3 in terms of our algorithm.

Let T denote the index set of a subset of the variables defining P. Then T will also denote a subsystem of the system S. Let $y_T = 0$ mean that all the variables in T have been set to 0. If we search for an extreme point of P with $y_T = 0$, there are two possibilities. The first is that we find an extreme point whose nonzero components index an I IS that does not intersect T. The second is that there is no such extreme point of P. In this case, the system $S \setminus T$ is feasible.

We can now modify step 3 of our algorithm (outlined above):

1. Identify an initial set of IISs J. (J may be empty.)

2. Solve the covering problem

$$\begin{array}{ll} \min & \sum_{i=1}^{m} c_i z_i \\ \text{subject to} & \\ & \sum_{i \in I} z_i \quad \geq 1 \quad \text{for all IISs } I \in J. \\ & z_i \text{ binary} \end{array}$$

   Let T index the elements in the optimal cover.

3. Look for an extreme point of P having $y_T = 0$. If there is none, STOP, T is an optimal cover of *all* IISs. Otherwise, add the IIS corresponding to this extreme point to J and go to 2.

This implementation is generalized in Section 4 to operate on an infeasible linear programming problem given in a more general format. Later sections discuss enhancements including the selection of the initial J in Step 1, and the choice of extreme point in Step 3. We also discuss the efficacy of using a heuristic to solve the covering problem in Step 2 when possible. Now, we demonstrate the utility of the IIS cover isolation.

3 Min IIS Cover Isolation in Practice

Finding a minimum weight IIS cover will take more time than identifying a single IIS and attempting infeasibility diagnosis. Is the extra information obtained worth the extra time expense? We address this by looking at several examples, and making a number of simple observations. Observe that if there exists a constraint which is contained in *every* IIS, it is a minimum IIS cover. So, in the *best* case, infeasibility can be isolated to a single constraint.

This is illustrated by the following example from [22]: Consider the infeasible system:

$$\begin{array}{llrcrcl} 1. & x_1 & - & x_2 & \leq & 0 \\ 2. & & & 2x_2 & \leq & 1 \\ 3. & x_1 & + & x_2 & \geq & 2 \\ 4. & & & x_2 & \geq & 2 \\ 5. & 2x_1 & + & x_2 & \geq & 4 \end{array}$$

An IIS of this system is {1,2,3}. Based upon this information alone, out of the context of the entire system, a modeler would have a difficult time determining the source of

the infeasibility. The unique min IIS cover is {2}, which indicates that every IIS in the system contains the second constraint.

On real problems, the isolation power becomes even greater (see Results section for a complete comparison). We look at a real industry example of using the min weight IIS cover to isolate infeasibility. This LP is a SONET network model having 845 rows, 1792 columns, and 8448 non-zero entries. The SONET model arises from a new family of problems in the telecommunications industry – it is a multidimensional packing problem with many complicating constraints. Specifically, there are a large number of capacity and demand constraints and a large number of side constraints. These side constraints are generated by a number of different software codes depending upon the exact model being analyzed. An instance of the model was found to be infeasible, and was analyzed using our algorithm. The min cardinality IIS cover isolated the infeasibility to a single demand constraint. Since this portion of the model was known to be valid, weights were assigned to rows of the infeasible problem and the min *weight* IIS cover was found. A weight of 1 was assigned to all software generated constraints, which accounted for 32% of all rows, and a weight of 5 was assigned to all capacity and demand constraints. These weights become the objective function value in the covering problem for each covering variable. The covering variables correspond to constraints of the original infeasible system. Resolving, we were able to isolate the infeasibility to a bad constraint. Looking at the code used to generate this constraint led to the discovery of a family of 5 bad constraints and correction of a section of code. It initially took 3.26 seconds on the RS6000 to determine that the problem was infeasible. Solving the min cardinality IIS cover took 7.28 seconds, and the final step of solving the min weight IIS cover took 7.21 seconds. Finding a single IIS using CPLEX run on the same machine took 86 seconds (after infeasibility was diagnosed), and yielded an IIS having 6 elements. This example provides an analysis path that has proven successful isolating infeasibility in other problems. Suppose a modeler takes the extra time during model development to generate weights for each constraint and includes these weights as an extra column in the LP. Then, when running the model, the bounds for this column can be set to zero (so it has no effect on the original LP model). If the model is found to be infeasible, the weights will be used by the min weight IIS cover algorithm to help isolate the infeasibility. As seen in this example, having constraint weights available can greatly increase the effectiveness of post-infeasibility analysis.

It is easily demonstrated however, that the modeling error is not always isolated by a particular min I IS cover. Any infeasible system having only a single I IS will have a min I IS cover consisting of *any* element of the I IS. By weighting the objective function of the covering problem, alternative minima can be identified to aid in isolation. This can be done by setting the objective function coefficients for variables contained in the original cover to a suitably large integer, say n – the number of variables in the covering problem. We then resolve the covering subproblem one time with the current I IS set. In summary, the min I IS cover *itself* will not always provide a perfect isolation to the modeling error; however, the isolation is always at least as useful as that obtained from a single I IS. Using our algorithm to find the min I IS cover always identifies at least one I IS. This combination of min I IS cover *and* I ISs is available to the user in her debugging analysis. Additionally, the I IS cover will *always* give the modeler the means of eliminating infeasibility from the model. A single I IS does not provide this level of isolation in general.

We present an example from the NETLIB infeasible library to demonstrate another key advantage of the min I IS cover. The WOODINFE problem is a small network example with 36 rows and 89 columns. An I IS infeasibility isolation from MINOS is a single row. However, the min I IS cover consists of 2 constraints. This problem has two disjoint modeling errors which **can not** both be identified by a single I IS. In infeasibility instances with multiple modeling errors the covering approach is especially powerful.

4 Formulation of P, the Alternative Polyhedron, for General Linear Programming Problems.

We now turn our attention back to the problem of identifying I ISs. Theorem 3 can be restated in a more general setting:

Theorem 3a: *Given the inconsistent system* $S = \{\mathbf{x} \in Q^n \mid A\mathbf{x} \leq \mathbf{b},\ C\mathbf{x} = d,\ \mathbf{L} \leq \mathbf{x} \leq \mathbf{U}\}$, *the indices of the minimal infeasible subsystems of* S *are exactly the supports of the vertices of the polyhedron* $P = \{\mathbf{y}, \mathbf{w}, \mathbf{v}, \mathbf{z} \in Q^m \mid \mathbf{y}^T A + \mathbf{w}^T C + \mathbf{v} - \mathbf{z} = 0,\ \mathbf{y}^T\mathbf{b} + \mathbf{w}^T\mathbf{d} + \mathbf{v}^T\mathbf{U} - \mathbf{z}^T\mathbf{L} = -1,\ \mathbf{y}, \mathbf{z}, \mathbf{v} \geq 0,\ \mathbf{w}$ *unrestricted*$\}$.

Note that if the only bounds on $\mathbf{x}$ are non-negativity constraints, we can simplify the above formulation:

Theorem 3b: *Given the inconsistent system* $S = \{\mathbf{x} \in Q^n \mid A\mathbf{x} \leq \mathbf{b},\ C\mathbf{x} = \mathbf{d},\ \mathbf{x} \geq$

$0\}$, *the indices of the minimal infeasible subsystems of* S *are exactly the supports of the vertices of the polyhedron* $P = \{\mathbf{y}, \mathbf{w} \in Q^m \mid \mathbf{y}^T A + \mathbf{w}^T C \geq 0,\ \mathbf{y}^T \mathbf{b} + \mathbf{w}^T \mathbf{d} = -1,\ \mathbf{y} \geq 0,\ \mathbf{w}$ *unrestricted*$\}$.

In other words, we do not need to explicitly handle non–negativity constraints, but can simply check the status of the slack variables of the alternative system to determine if non–negativity of some x_i is included in an IIS. Also, this helps reduce the size of the LP instance we solve at each step.

If the right hand side of the "cone–bounding" constraint were left as -1 for large problems, numerical instability would result. We observed on several problems that when we fixed at 0 those variables in the alternative system which are in the covering solution, we obtained an IIS identical to the one just previously generated. But this means that in trying to find an IIS not covered by the current solution, we have generated one that *is* covered by the current solution. Examine what can happen if P has many columns. Suppose each variable in the basis is set within epsilon of 0 and the constraint $\mathbf{y}^T \mathbf{b} + \mathbf{w}^T \mathbf{d} = -1$ is satisfied. However, these values *individually* fall below the optimizer's zero tolerance, so under ideal conditions, the $\mathbf{y}^T A + \mathbf{w}^T C \geq 0$ constraints will also hold, and so a "numerically" feasible solution will be obtained by the optimizer for P. However, since the values of the basic variables are below the software imposed tolerance, they should be considered 0, and so we should have that $\sum(y_i * k_i) + \sum(w_j * l_j) = 0$ for any $\mathbf{k}, \mathbf{l}$; which means that the solution is **not** basic feasible. When this "solution" is identified as being feasible, then we identify subsystems which are **not** IISs. We alleviate this problem by setting the right hand side of this constraint to be equal to the number of variables in the problem. Although theoretically we need a larger value than this, this fix appears to have resolved the problem.

5 Finding IISs

Two slightly different approaches may be used to identify IISs of S. First, we consider using the simplex algorithm to find extreme points of P. By bounding the objective function $\mathbf{y}^T \mathbf{b}$ of the alternative system and including it in the constraint set of P, we free ourselves to use a "surrogate" objective function to heuristically guide our search. We have experimented with using $\mathbf{1}$ as the objective weight. This should tend to find small cardinality IISs, although to find the true minimum cardinality IIS we must solve a fixed–charge integer programming problem. Additionally, these weights

can be modified at each step by incrementing the weight of each variable appearing in the current IIS. In this way, we heuristically tend to find IISs which overlap in the fewest number of elements.

We have also considered ignoring the objective function when looking for extreme points of P and using the first feasible solution found. Since every extreme point corresponds to an IIS, we still identify an IIS. Typically this takes fewer pivots to find an IIS than solving to optimality with the surrogate objective described above. However, the savings in IIS identification time may be offset by the typically larger cardinality IISs found and the increased number of covering subproblems needed to solve.

Rather than generating IISs individually, we can generate many by solving a single LP if we instead search for extreme rays on the cone P'. We consider the following linear programming instance based upon our definition of P':

$$\begin{array}{lll} \min & \mathbf{y}^T\mathbf{b} & \\ \text{subject to} & & \\ & \mathbf{y}^T A & = 0 \\ & \mathbf{y} & \geq 0 \end{array}$$

Since S is infeasible, we know that P' is feasible (and hence unbounded). Therefore, we must be able to find at least one nonbasic variable of non-optimal sign at termination of our new LP. If this nonbasic is not blocked, then the variable and all basic variables which correspond to non–zero tableau column entries for the nonbasic form the support set of an IIS. In other words, the supports of the extreme rays of P' are IISs in the original infeasible system, S. In general, we will have more than a single such nonbasic variable, so we can find many IISs from one LP solution. By pivoting we can find all IISs of S by such a procedure.

In general, we will have to perform several pivots before identifying unboundedness due to the degeneracy of the $\mathbf{0}$ solution. In practice, this method can reduce the number of iterations to solve the minimum weight IIS cover. For example, let an IIS of S be identified as {1,2,3,4,5,8,10,15} and suppose that constraint 15 is in *every* IIS (and so is the minimum IIS cover). If {1} is identified as the cover, there could exist another IIS of the form {2,3,4,5,6,8,10,15} and so forth. There could be potential to identify *many* IISs before the min IIS cover is solved optimally. If a conical solution yields multiple IISs, the total number of covering subproblems (and hence IIS identification problems) can be reduced. This becomes important for large infeasible problems which can have a single constraint as the optimal min IIS cover,

while having IISs with hundreds of elements in them.

This idea has even broader implications. We are looking for extreme rays of the alternative system. The alternative system is simply the dual of our original (primal) system with the primal objective function zeroed out. Thus, by ignoring the objective function, the Phase 1 solution of S can be used to identify an extreme ray of the unbounded dual system, or an IIS of the original system. In other words, we can identify one (or more) IISs with no more work than solving phase 1 of our original system. In essence, this generalizes the result of Van Loon [22]. Assume that we are working with an infeasible system $A\mathbf{x} \leq \mathbf{b}$, where any non–negativity constraints are considered for this discussion to be included. Equivalently, we have the infeasible system $A\mathbf{x} + \mathbf{s} = \mathbf{b}$, where $\mathbf{s}$ is a vector of slack variables. Formulate the following LP:

$$\begin{array}{lll} \max & \mathbf{c}^T\mathbf{x} & \\ \text{subject to} & & \\ & A\mathbf{x} + \mathbf{s} & = \mathbf{b} \\ & \mathbf{s} & \geq 0 \end{array}$$

Suppose that at the termination of Phase 1 for the infeasible system we have a basic slack variable, say s_1, which is strictly less than 0. Additionally, if the Phase 1 tableau entries for all nonbasic variables x_i are 0 and for all nonbasic slack variables s_i are non–negative in the row corresponding to s_1, then Van Loon proves the following:

Theorem 5 ([22]): *After deletion of all constraints $s_j \geq 0$ for which their corresponding tableau entries in the row corresponding to basic variable s_1 are 0, the subsystem $\{s_1 \geq 0,\ s_j \geq 0 \mid$ all remaining non-basic variables $s_j\}$ is an IIS*

Van Loon's algorithm, without explicitly identifying it as such, finds extreme rays of the alternative system by pivoting through bases of the original infeasible system looking for tableau rows satisfying the above theorem.

Since solving on the cone of the alternative system allows us the potential to identify multiple IISs from a single basis and in general takes fewer pivots to solve, the idea is to use this as a "jump–start" for the algorithm, and then to use the extreme point method to find IISs with specific characteristics.

We can illustrate each of these methods using the following example from [5] using

LINDO. Suppose we are given the infeasible system S defined as:

$$
\begin{array}{llrcrcrcrl}
1. & -0.5x_1 & + & x_2 & & & & & \geq & 0.5 \\
2. & 2x_1 & - & x_2 & & & & & \geq & 3.0 \\
3. & 3x_1 & + & x_2 & & & & & \leq & 5.0 \\
4. & & & & & & & x_5 & \leq & 2.0 \\
5. & & & & & 3x_4 & - & x_5 & \leq & 2.0 \\
6. & & & & & x_4 & & & \geq & 5.0 \\
7. & x_1 & & & & & + & x_5 & \leq & 10.0 \\
8. & x_1 & + & 2x_2 & + & x_4 & & & \leq & 14.0 \\
9. & & & x_2 & + & x_4 & & & \geq & 1.0
\end{array}
$$

with all $x_i \geq 0$

If we solve to optimality using objective weights **1** for all variables of the alternative system, after 6 pivots, we find {4,5,6} as an IIS. If we solve to feasibility only (ignore the objective weights), we find the IIS {1,2,3} after 5 pivots. Solving on the cone we find 3 IISs after only 4 pivots: {4,5,6}, {1,2,3}, and {1,2,5,6,7}. The min IIS cover is {1,5}, so by solving on the cone, we initialize our procedure with enough IISs to solve the cover problem only once.

6 Heuristic Solution of Covering Problems

The covering problems can be solved heuristically at intermediate steps. We can determine if these suboptimal covers are true covers of the min weight covering problem just as we did when solving this problem optimally in Step 3 of our algorithm. If a true optimal cover is desired, the covering problem must be solved optimally before terminating.

The greedy heuristic [6] can be used at intermediate steps. When no IIS is found that is not covered by the greedy solution, we solve the covering problem optimally. If the cover is the same cardinality as that found by the greedy heuristic, we are done. Otherwise, we find an IIS not covered by the current solution and continue the algorithm. The greedy algorithm will not guarantee an optimal cover; however, it seems to work well in practice.

For the problems in the infeasible library on NETLIB, we have found that typically only a small fraction of total algorithm time (usually less than 10%) is spent solving the covering subproblems. This result is based on identifying IISs singly by searching for extreme point solutions of P, so the results could change significantly with improvements over the basic form of the algorithm. It should be noted that the difficulty of the covering problem has been displayed on a few relatively small

problems. For the NETLIB problem MONDOU2, more than 97% of the CPU time (115 of 118 seconds) was spent solving a *single instance* of the covering subproblem having only 9 constraints. This demonstrates a need to explore more fully using the greedy algorithm, or a modification of it, at intermediate steps.

7 Selection of Violated I IS

As discussed earlier, if a simplex-like procedure is used to find the I IS in Step 3 of the algorithm, we can steer the procedure to find an I IS having certain desirable characteristics. If we weight all variable coefficients of the objective with 1 and find a minimum weight extreme point, we will tend to find a *small* cardinality I IS. If we weight each variable according to how many generated I ISs it already appears in, we will tend to find an I IS that overlaps as little as possible with the ones we already have. Both these strategies have intuitive merit with respect to reducing the number of iterations of constraint generation described. A comparison of these ideas is included in the results section.

8 Computational Results

All comparisons (unless otherwise stated) were run on an IBM RS6000 using the IBM OSL object library and FORTRAN code written by the authors. All runs were made with approximately the same computer load – only a single user was on the system. We experimented with three versions of our algorithm. All three solve the covering subproblem to optimality at each step using the OSL EKKMSSL subroutine. Due to coding complications involved in solving on the cone, all three versions presented here search for extreme point solutions on the bounded alternative system. We distinguish each version by the objective function used in solving the alternative system. The baseline algorithm, IIS_COV_1, weights the coefficients of the objective function to all 1's in an attempt to find small cardinality I ISs. Version 2, IIS_COV_2, weights the coefficient of each variable according to how many generated I ISs it already appears in. Intuitively, this will tend to find I ISs which overlap as little as possible. Finally, version 3, IIS_COV_3, ignores the objective function, and the first extreme point found is the solution. This will tend to identify I ISs faster, though perhaps at the trade off of finding larger cardinality I ISs. Note that IIS_COV_2 will differ from IIS_COV_1 only when more than 1 I ISis found to optimally solve the covering problem. Therefore,

we did not run IIS_COV_2 if we found that only 1 IIS was needed to find the optimal cover.

The test bed of problems consists of the infeasible LP library on NETLIB, originally set up by John Chinneck. A summary of the problems is given in Table 1.

Problem	Original Problem			Alternative System		
	Constraints	Variables	Nonzeros	Constraints	Variables	Nonzeros
BGDBG1	348	407	1440	408	390	1737
BGETAM	400	688	2409	689	689	2957
BGINDY	2671	10116	65502	10117	2671	67589
BGPRTR	20	34	64	35	20	76
BOX1	231	261	651	262	492	1173
CERIA3D	3576	824	17602	825	3576	17851
CHEMCOM	288	720	1566	721	625	2180
CPLEX1	3005	3221	8944	3222	3223	10883
EX72A	197	215	467	216	412	897
EX73A	193	211	457	212	404	879
FOREST6	66	95	210	96	71	226
GALENET	8	8	16	9	16	38
GOSH	3792	10733	97231	10734	3792	97433
GREENBEA	2393	5405	30885	5406	2941	31926
ITEST2	9	4	17	5	9	26
ITEST6	11	8	20	9	11	31
KLEIN1	54	54	696	55	54	700
KLEIN2	477	54	4585	55	477	4600
KLEIN3	994	88	12107	89	994	12115
MEXP	1383	1500	5027	1501	1383	5755
MONDOU2	312	604	1208	605	1520	3088
PANG	361	460	2652	461	453	2587
PILOT4I	410	1000	5141	1001	717	5860
QUAL	323	464	1646	465	835	2662
REACTOR	318	637	2420	638	932	3699
REFINERY	323	464	1626	465	835	2641
VOL1	323	464	1646	465	835	2662
WOODINFE	35	89	140	90	69	208

Table 1. Problem Test Bed

In order to facilitate the analysis of results, we partitioned the problems into 3 sets according to their size. We considered "small" problems to be those having fewer than 100 rows and 100 columns. "Medium" problems are those with between 100 and 1000 rows and 100 to 1000 columns. The remainder are "large" problems.

Table 2 presents the results of our 3 approaches on the small problems. The measures we use in comparing the approaches are total solution time of the covering problem (CPU seconds), number of IISs found in solving the covering problem optimally, number of pivots to first IIS, average number of pivots per IIS, and the smallest cardinality IIS. In analyzing infeasibility, it may be useful to measure the size of an IIS in terms of the number of actual *constraints* it contains, rather than constraints *and* variable bounds. Note that IIS size refers to the number of constraints and

variable bounds *exclusive of non–negativity bounds.* Information on non–negativity bounds is available with the solution approach, we have chosen to not include it in the IIS reporting. The columns for average IIS size and smallest cardinality IIS are thus subdivided into IIS size and number of actual constraints.

On most problems, very little time is spent in solving the min weight covering subproblem, and from 50% to 95% of the time is spent in identifying IISs – solving the alternative system LP. Therefore, another reasonable measure of how well each algorithm performs is the number of simplex pivots required to find an IIS.

Problem	Algorithm	Cover Time	# IISs	Pivots		Avg. IIS		Smallest IIS	
				IIS 1	Avg.	Size	Rows	Size	Rows
BGPRTR	IIS_COV_1	0.19	1	18	10.0	7.0	7.0	7	7
BGPRTR	IIS_COV_3	0.20	1	14	8.0	14.0	14.0	14	14
FOREST6	IIS_COV_1	0.73	2	124	50.3	64.0	59.0	64	59
FOREST6	IIS_COV_2	0.71	2	124	51.0	64.0	59.0	64	59
FOREST6	IIS_COV_3	0.61	2	81	29.3	69.5	64.5	69	64
GALENET	IIS_COV_1	0.19	1	7	3.5	6.0	3.0	6	3
GALENET	IIS_COV_3	0.20	1	5	3.0	6.0	3.0	6	3
ITEST2	IIS_COV_1	0.38	3	4	2.0	3.7	3.7	3	3
ITEST2	IIS_COV_2	0.35	3	4	2.0	3.7	3.7	3	3
ITEST2	IIS_COV_3	0.36	3	4	1.8	3.7	3.7	3	3
ITEST6	IIS_COV_1	0.65	3	6	2.8	3.3	3.3	3	3
ITEST6	IIS_COV_2	0.69	3	9	4.0	3.3	3.7	3	3
ITEST6	IIS_COV_3	0.65	6	3	1.7	3.7	3.7	3	3
KLEIN1	IIS_COV_1	0.89	1	162	81.5	51.0	51.0	51	51
KLEIN1	IIS_COV_3	0.82	1	116	71.5	51.0	51.0	51	51
WOODINFE	IIS_COV_1	0.46	2	24	12.6	2.0	1.0	2	1
WOODINFE	IIS_COV_2	0.41	2	24	12.6	2.0	1.0	2	1
WOODINFE	IIS_COV_3	0.48	2	9	11.0	2.0	1.0	2	1

Table 2. Results for Small Problems

The first few pivots made by OSL during Phase 1 will be random. This is done initially to speed up the simplex algorithm, and later other pricing schemes are used. This means that when re-running a problem with the same algorithm we will have variance in our solution times. On the problems we looked at, this variance was typically 5% or less. For this reason, we will consider time differences between algorithms to be significant only if they are greater than 10%. In this light, no significant time difference is found between the algorithms on the small problem test bed. On the 4 problems where all 3 algorithms were run, IIS_COV_2 was faster on 2, IIS_COV_3 and IIS_COV_1 were faster on 1 each, with all three indistinguishable on one. If we try to verify our assumptions on strengths of each algorithm, we see that IIS_COV_3 takes fewer pivots both to find the first IIS and on average; but requires more IISs to solve the cover problem only on ITEST6. IIS_COV_1 does find smaller IISs on average than the other algorithms, however, not much more information can be obtained from the

small problems.

Algorithmic comparisons for the "mid–sized" problems are presented in Table 3. We have several interesting problems in terms of the size of the min weight IIS covering problem solved. With these harder problems, we see a number of interesting trends. In 10 of the 12 problems, IIS_COV_3 takes fewer pivots to find an initial IIS as expected. However, this algorithm has the lowest average number of pivots per IIS in only 7 of the 12 problems. What we see is that both IIS_COV_1 and IIS_COV_2 typically take more pivots to find an initial solution. However, only a few pivots, typically fewer than 25, are needed to find IISs satisfying optimality conditions for the alternative system after the initial basic feasible solution is found. Thus some of the advantage of IIS_COV_3 is lost when finding many IISs. In fact, on those problems where IIS_COV_3 took fewer pivots to find its initial IIS but had a higher average pivot value, an algorithm needing to find a larger number of IISs to solve the covering problem had the lowest average pivot value. Thus the more IISs one needs to find, the less important the initial number of pivots becomes in terms of solution time.

Additionally, we see that IIS_COV_3 solves the covering problem faster on 8 of the 9 problems with time distinctions. We also note that it finds the smallest average IIS on 3 of the 9 problems where distinction is made, and finds more IISs in solving the cover on only 4 problems and finds the least IISs in solving the cover 5 times (excluding ties). This seems to contradict the intuition that we would find IISs faster, but require more IISs to solve the covering problem.

Problem	Algorithm	Cover Time	# IISs	Pivots		Avg. IIS		Smallest IIS	
				IIS 1	Avg.	Size	Rows	Size	Rows
BGDBG1	IIS_COV_1	8.05	33	112	12.4	8.1	7.4	2	2
BGDBG1	IIS_COV_2	6.15	24	112	14.6	7.0	6.5	2	2
BGDBG1	IIS_COV_3	7.74	36	34	8.6	6.9	6.1	2	2
BGETAM	IIS_COV_1	10.42	5	242	206.8	68.0	62.2	8	7
BGETAM	IIS_COV_2	11.79	6	150	145.4	59.5	51.7	13	12
BGETAM	IIS_COV_3	9.24	7	20	89.1	173.1	140.7	18	17
BOX1	IIS_COV_1	1.46	2	13	36.0	53.5	52.5	10	9
BOX1	IIS_COV_3	1.46	2	12	34.3	54.5	53.5	10	9
CHEMCOM	IIS_COV_1	3.74	2	177	136.0	26.0	16.0	25	15
CHEMCOM	IIS_COV_2	3.73	2	177	142.0	24.0	14.0	23	13
CHEMCOM	IIS_COV_3	3.63	1	137	204.5	27.0	15.0	27	15
EX72A	IIS_COV_1	1.35	3	104	27.0	69.0	68.0	59	58
EX72A	IIS_COV_2	1.66	3	104	39.5	70.3	69.3	61	60
EX72A	IIS_COV_3	1.22	2	108	36.0	66.5	65.5	59	58
EX73A	IIS_COV_1	0.92	1	88	44.0	28.0	27.0	28	27
EX73A	IIS_COV_3	0.97	1	79	42.5	26.0	25.0	26	25
KLEIN2	IIS_COV_1	8.03	13	597	85.4	53.2	53.2	53	53
KLEIN2	IIS_COV_2	5.46	4	574	173.8	54.0	54.0	53	53
KLEIN2	IIS_COV_3	5.38	6	476	113.9	55.2	55.2	53	53
PANG	IIS_COV_1	3.21	1	392	204.5	16.0	13.0	16	13
PANG	IIS_COV_2	2.59	1	250	127.5	16.0	13.0	16	13
PANG	IIS_COV_3	2.65	2	183	76.3	19.5	16.0	18	15
QUAL	IIS_COV_1	8.05	5	707	149.3	185.4	124.0	142	92
QUAL	IIS_COV_2	12.12	5	707	223.2	184.8	103.6	143	82
QUAL	IIS_COV_3	7.40	6	558	107.9	180.6	124.0	134	90
REACTOR	IIS_COV_1	5.58	2	152	192.0	5.0	1.0	5	1
REACTOR	IIS_COV_2	4.44	2	152	125.7	5.0	1.0	5	1
REACTOR	IIS_COV_3	3.89	2	100	119.3	5.0	1.0	5	1
REFINERY	IIS_COV_1	22.26	38	665					
REFINERY	IIS_COV_2	19.99	31	665	48.1	134.0	89.0	93	63
REFINERY	IIS_COV_3	16.51	23	577	72.6	145.7	93.4	97	62
VOL1	IIS_COV_1	13.45	13	699	83.3	180.8	121.7	137	91
VOL1	IIS_COV_2	20.70	10	699	278.5	180.3	121.6	137	91
VOL1	IIS_COV_3	10.75	8	715	138.6	182.6	120.9	160	104

Table 3. Results for Mid–sized Problems

As we look at the results for the large problems (Table 4), we see that an undirected search for IISs is faster. IIS_COV_3 again dominates, being the fastest algorithm on 8 of the 9 problems, taking the fewest pivots to the first IIS on all problems, and taking fewest average number of pivots on 6 of 9 problems. It also finds the smallest average IIS on 4 of 6 problems where distinction is made.

Problem	Algorithm	Cover Time	# IISs	Pivots		Avg. IIS		Smallest IIS	
				IIS 1	Avg.	Size	Rows	Size	Rows
BGINDY	IIS_COV_1	139.19	1	1908	1140.0	3.0	3.0	3	3
BGINDY	IIS_COV_3	126.34	1	219	1114.0	3.0	3.0	3	3
CERIA3D	IIS_COV_1	56.32	3	1905	758.0	123.3	123.3	73	73
CERIA3D	IIS_COV_2	66.46	3	2406	921.0	141.3	141.3	74	74
CERIA3D	IIS_COV_3	26.45	8	538	99.0	144.8	144.8	73	73
CPLEX1	IIS_COV_1	38.50	1	1473	1214.5	5.0	5.0	5	5
CPLEX1	IIS_COV_3	33.83	1	212	1058.5	5.0	5.0	5	5
GOSH	IIS_COV_1	563.91	1	6009	3305.5	49.0	49.0	49	49
GOSH	IIS_COV_3	637.34	2	1885	2350.0	72.0	72.0	68	68
GREENBEA	IIS_COV_1	410.50	3	2983	1955.8	9.0	6.3	4	2
GREENBEA	IIS_COV_2	359.95	4	2983	1319.6	17.8	15.0	4	2
GREENBEA	IIS_COV_3	322.34	3	10	1442.8	8.3	5.3	4	1
KLEIN3	IIS_COV_1	97.16	29	1571	506.0	101.7	101.7	95	95
KLEIN3	IIS_COV_2	49.77	12	1560	538.8	87.4	87.4	86	86
KLEIN3	IIS_COV_3	36.34	21	1137	205.6	85.8	85.8	84	84
MEXP	IIS_COV_1	12.37	2	377	157.0	8.0	8.0	7	7
MEXP	IIS_COV_3	8.76	1	176	237.5	6.0	6.0	6	6
MONDOU2	IIS_COV_1	17.69	14	503	60.8	163.5	146.6	22	17
MONDOU2	IIS_COV_2	118.63	13	503	58.1	182.3	163.8	22	17
MONDOU2	IIS_COV_3	17.11	9	460	62.6	146.9	128.9	22	17
PILOT4I	IIS_COV_1	28.82	1	659	930.0	1.0	1.0	1	1
PILOT4I	IIS_COV_2	18.05	1	1010	629.0	1.0	1.0	1	1
PILOT4I	IIS_COV_3	7.20	1	86	202.5	1.0	1.0	1	1

Table 4. Results for Large Problems

We also compare (Table 5) the information provided by the IIS cover to that obtained from IIS isolation via MINOS (from [4]). Many of the problems in this test bed were created by taking a feasible LP instance and modifying a single bound or constraint until the problem was infeasible. Four of the problems in the test bed we know to be infeasible in original form – BGDBG1, BGPRTR, GREENBEA, and MONDOU2. Of these problems, only BGPRTR has an IIS cover of 1 (and requires a single IIS to solve the cover). Thus although there are a large number of problems in the test bed having singleton IIS covers and requiring only a single IIS to find their cover, this may be an artifact of their construction. We feel that the min weight IIS cover will be a valuable tool in debugging LPs at the model development and integration stage.

Problem	MINOS(IIS)	Min IIS Cover		Smallest IIS	
	Rows	Cardinality	IISs Found	Size	Rows
BGDBG1	2	12	38	2	2
BGETAM	7	1	5	8	7
BGINDY		1	1	3	3
BGPRTR	5	1	1	7	7
BOX1	8	1	2	10	9
CERIA3D	73	1	3	73	73
CHEMCOM	7	1	2	23	13
CPLEX1	5	1	1	5	5
EX72A	58	1	3	59	58
EX73A	24	1	1	26	25
FOREST6	55	1	2	64	59
GALENET	2	1	1	6	3
GOSH		1	1	49	49
GREENBEA	1	2	3	4	1
ITEST2	3	2	3	3	3
ITEST6	2	2	3	3	3
KLEIN1	50	1	1	51	51
KLEIN2	51	1	4	53	53
KLEIN3	74	1	21	84	84
MEXP	5	1	1	6	6
MONDOU2	15	3	9	22	17
PANG	11	1	1	16	13
PILOT4I	1	1	1	1	1
QUAL	76	1	6	134	90
REACTOR	1	1	2	5	1
REFINERY	47	1	38	92	61
VOL1	80	1	10	131	91
WOODINFE	1	2	2	2	1

Table 5. Min IIS Cover and MINOS IIS Isolation Comparison

9 Conclusions and Avenues for Further Research

We are currently working on implementing a version of our algorithm to solve on the cone directly (either in the alternative system, or in the original infeasible system). We feel that the results obtained thus far are very promising, and that we can speed up the IIS identification procedure by solving on the cone to provide multiple IISs for the initial covering subproblem.

Additionally, the extra time spent heuristically attempting to find small cardinality IISs or minimally overlapping IISs does not appear to make the problems any easier to solve. A method of making these techniques more viable would be to use every IIS found while attempting to find the optimal extreme point of the alternative system. Currently, only the IIS found at the final extreme point solution is added to the covering problem during each IIS identification phase.

We have also developed a version of the code which initially finds all disjoint IISs

before solving the cover. The motivation for pursuing this algorithm is to quickly find a good bound on the size of the cover. Experiments with this code also look promising.

We are also examining the use of heuristics to solve the covering problem at intermediate steps. At the present time, this is of secondary concern. We are focusing more attention on the theoretical structure of the min weight I IS covering problem. We are attempting to identify facets of this problem which can be included in a branch and cut type algorithm.

Additionally, since the linear separator or linear discriminant problem can be recast as the problem of finding a maximum feasible subystem of an infeasible linear program, we have been exploring the use of this algorithm to solve this and other related problems.

10 Acknowledgements

The authors wish to thank Harvey Greenberg for his many discussions on the topic and for several algorithmic recommendations. We also wish to thank John Chinneck for informing us of the existence of the NETLIB infeasibility library and providing comments on draft versions of this paper.

References

[1] Berge C., *Hypergraphs*, North-Holland, Amsterdam, 1989.

[2] Chakravarti N., Three Approaches to Post-Infeasibility Analysis, WPS-147(90), Indian Institute of Management, 1990.

[3] Chinneck J., Finding Minimal Infeasible Sets of Constraints in Infeasible Mathematical Programs, Technical Report SCE-93-01, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1993.

[4] Chinneck J., Finding the Most Useful Subset of Constraints for Analysis in an Infeasible Linear Program, Technical Report SCE-93-07, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1993.

[5] Chinneck J. and Dravnieks E., Locating Minimal Infeasible Constraint Sets in Linear Programs, *ORSA Journal on Computing* Vol. 3 (1991), 157-168.

[6] Chvátal V., A Greedy Heuristic for the Set Covering Problem, *Mathematics of Operations Research*, Vol. 4 (1979), 233-235.

[7] Fan K., On Systems of Linear Inequalities, in *Annals of Mathematical Studies Number 38: Linear Inequalities and Related Systems*, edited by Kuhn H.W. and Tucker A.W., Princeton University Press, Princeton, NJ., 1956, pp. 99-156.

[8] Gary M. and Johnson D., *Computers and Intractability*, W.H. Freeman, New York, 1979.

[9] Gleeson J. and Ryan J., Identifying Minimally Inconsistent Subsystems of Inequalities, *ORSA Journal on Computing*, Vol. 2 (1990), 61-63.

[10] Greenberg H.J., Diagnosing Infeasibility for Min-Cost Network Flow Models, Part I: Dual Infeasibility, *IMA Journal of Mathematics in Management*, Vol. 1 (1987), 99-110.

[11] Greenberg H.J., Diagnosing Infeasibility for Min-Cost Network Flow Models, Part II: Primal Infeasibility, *IMA Journal of Mathematics in Business and Industry*, Vol. 4 (1988), 39-50.

[12] Greenberg H.J., An Empirical Analysis of Infeasibility Diagnosis for Instances of Linear Programming Blending Models, *IMA Journal of Mathematics in Business and Industry*, Vol.4, 163-210.

[13] Greenberg H.J., How to Analyze Results of Linear Programs, Part 3: Infeasibility Diagnosis,*Interfaces*, Vol. 23, No. 6 (1993).

[14] Greenberg H.J., Consistency, Redundancy, and Implied Equalities in Linear Systems, draft version 1993.

[15] Greenberg H.J. and Murphy F.H., Approaches to Diagnosing Infeasibility for Linear Programs, *ORSA Journal on Computing*, Vol. 3, No. 3, (1991), 253-261.

[16] Motzkin T.S., Contributions to the Theory of Linear Inequalities, Doctoral Dissertation, University of Basel, 1933, translated by D.R. Fulkerson, in *Theodore S. Motzkin: Selected Papers*, edited by D. Cantor, B. Gordon, and B. Rothschild, Birkhauser,1983.

[17] Nemhauser G. and Wolsey L., *Integer and Combinatorial Optimization*, John Wiley and Sons, 1988.

[18] Ryan J., Transversals of IIS–Hypergraphs, *Congressus Numerantium*, Vol. 81 (1991), 17-22.

[19] Ryan J., IIS–Hypergraphs, draft version 1994.

[20] Sankaran J.K., A Note on Resolving Infeasibility in Linear Programs by Constraint Relaxation, *Operations Research Letters*, Vol. 13 (1993), pp. 19-20.

[21] Schrijver, A., *Theory of Linear and Integer Programming*, John Wiley and Sons, 1986.

[22] van Loon J., Irreducibly Inconsistent Systems of Linear Inequalities, *European Journal of Operations Research*, Vol. 8 (1981), pp. 283-288.