INFORMS Journal on Computing
Vol. 8, No. 1, Winter 1996

0899-1499/96/0801-0055 $01.25
© 1996 INFORMS

Localizing and Diagnosing Infeasibilities in Networks

John W. Chinneck / *Systems and Computer Engineering, Carleton University, Ottawa, Ontario K1S 5B6, Canada; Email: chinneck@sce.carleton.ca*

(Received: December 1993; revised: June 1995; accepted: August 1995)

Network models are among the largest linear programs solved, but formulation can be a bottleneck. Errors may be introduced during formulation, reformulation, or when combining several smaller models into one large model (as is often done in econometrics). Simply knowing that the model is infeasible is not enough: where there are many nodes and arcs, the modeler needs guidance as to where repairs are needed (localization), and some indication of the cause of the error (diagnosis). The state of the art flow balancing methods for analyzing infeasible networks simply do not provide enough localization or diagnosis, and in addition cannot operate on advanced netforms. The paper presents an infeasibility analysis procedure for all classes of network models that localizes the error to a minimal causative set of nodes, arcs, and constraints, and provides a basic diagnosis. The procedure builds on previous work[3, 5, 6] but tunes the methods for the special case of networks. Examples are given, using solutions provided by PROFLOW, software for the formulation, analysis and solution of network models.

Because the ability to solve very large linear programs is now widely available, the main burden in linear program (LP) modeling has shifted to the initial formulation of a correct model. Errors may be introduced during initial formulation, during reformulation, or during the combination of several small models into a single large model. Since the models are so large, automated formulation assistance is required, and the need is especially acute in the case of network LPs, which are among the largest optimization models known.

We concentrate here on analyzing infeasibility in network models, a serious and all-too-common class of formulation error. The goal is to develop methods that *localize* the infeasibility to a small part of the network and that suggest a *diagnosis* of the error so that a repair can be effected.

Existing methods specifically for analyzing infeasible networks rely exclusively on the venerable supply and demand balancing procedures developed by Gale,[12] Fulkerson,[11] Hoffman,[22] and Ford and Fulkerson.[10] These procedures have important disadvantages: (1) they often provide insufficient localization of the infeasibility, and (2) they apply only to simple pure networks and so cannot analyze more advanced netforms.

More recent methods for analyzing infeasibility in general LPs[6–8, 13] of course apply to all network forms and are an important part of the procedure developed here. However, better diagnostics are provided by combining these general methods with a technique for analyzing networks for another type of error (nonviability) in an ordered set of tests. The procedure developed herein addresses the deficiencies of the supply-demand balancing approaches. It provides a better localization of the problem and an automated basic diagnosis, and applies to all netforms, including generalized and processing networks, and networks with extra side constraints.

Examples are given to demonstrate the effectiveness of the procedure. The calculations are carried out by PROFLOW, a computer code developed by the author to formulate, analyze, and solve network models of all types.

1. A Taxonomy of Network Models

The most basic form of a network model is a *pure network* in which all nodes either conserve flow (this is a *regular node*) or are sources or sinks. Arcs conduct flow between nodes and have a *nonnegativity constraint* as the lower flow bound and an infinite upper bound, or possibly a positive lower and/or upper flow bound. Arcs may also have an associated cost per unit of flow. The idea is normally to minimize the total cost of the flow while satisfying all constraints.

A *generalized network* includes at least one *generalized arc*. A generalized arc has an associated *multiplier* or *gain* that multiplies the flow into the tail of the arc to yield the flow out of the head of the arc. See [15] for details on modeling using generalized networks.

A *processing network* contains at least one *processing node*. A processing node is constrained by fixed proportions of the flows in the arcs incident on it (Figure 1). This is particularly useful for modeling engineering or economic processes. Simple examples include a food preparation process in which fixed proportions of ingredients are combined into an output product, or an oil refining process in which a crude oil feedstock is broken into fixed proportions of output products. In a *pure processing network* all of the processing nodes conserve flow as well as enforce the fixed proportions. In a *nonconserving (NC) processing network*, at least one of the processing nodes (called a *nonconserving (NC) processing node*), does not conserve flow (the sum of the inflow proportions does not equal the sum of the outflow proportions). See [3, 5, 23] for details on modeling using processing networks.

Note that a generalized network is easily converted to a NC processing network by replacing each generalized arc by two arcs with an intervening NC processing node and appropriate processing flow proportions.[5]

Subject classifications: Programming: linear: algorithms, theory. Network/graphs: theory.

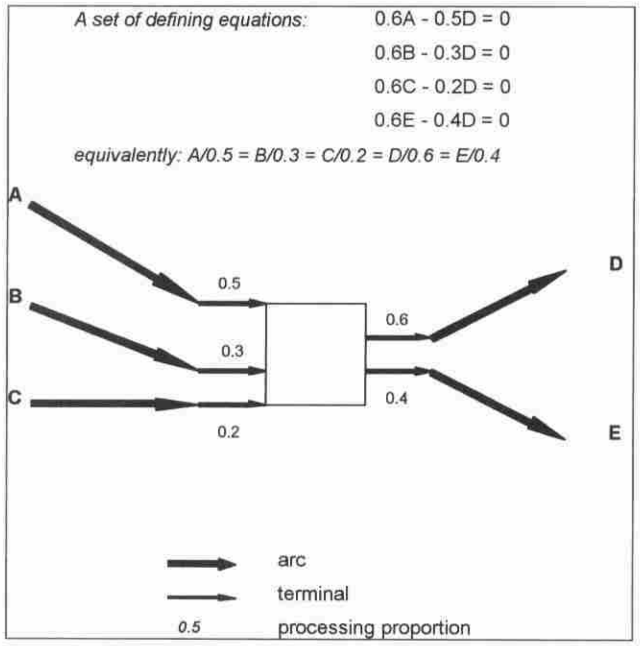

Figure 1. Example of a processing node.

It is also important to distinguish between the classes of constraints appearing in a network model:

Structural constraints include (1) the flow conservation equation for regular nodes, (2) a sufficient set of proportion equations for processing nodes of both types,[3] and (3) the arc flow nonnegativity constraints.

Flow bounds are the simple nonnegative upper or lower bounds on the flow in an arc. Negative flow bounds are not allowed. Unless otherwise stated, the lower flow bound is assumed to be zero and the upper flow bound to be infinity.

Extra side constraints are any other constraints linking the arc flow variables. Generalized upper or lower bounds and other arbitrary constraint are in this category. "Side constraint" is a generic term denoting any constraint that is not a simple flow conservation equation or a simple upper or lower arc flow bound. Note the distinction here between two subsets of side constraints: "processing node proportion equations" and "extra side constraints."

2. Supply-Demand Flow Balancing

As described by Greenberg and Murphy,[20] the earliest theorems used for analyzing infeasible pure networks deal with supply and demand balances, including those by Gale,[12] Fulkerson,[11] Hoffman,[22] and Ford and Fulkerson.[10] As Greenberg and Murphy point out, the guidance provided by algorithms relying on these theorems is very often insufficient to clearly identify the cause of the problem. More exact localization is needed.

Hao and Orlin[21] apply the Gale-Fulkerson-Hoffman theorems in a maximum flow algorithm procedure for identifying a "witness" set of nodes for which the net supply and the total outflow capacity conflict. A useful feature is a procedure for finding a minimal witness set.

Greenberg[17, 18] provides additional localization through a set of sophisticated heuristics derived by combining the flow balance theorems and logic about network behaviour (e.g. path and cycle generation); this has been implemented in the ANALYZE software.[19] When certain conditions indicating infeasibility are recognized, tracing back through the series of manipulations helps to localize the cause. At their best, Greenberg's algorithms provide an effective localization and diagnosis of the network infeasibility. At worst, the localization provided by the basic flow balance theorems is guaranteed.

The state-of-the-art algorithms specifically for the analysis of infeasible networks all rely on flow balancing, which has two important drawbacks. First, the diagnosis often fails to provide a useful localization (though Hao and Orlin's minimal witness set may help), and second, flow balancing does not work on netforms beyond pure networks, for example on generalized or processing networks. New methods are needed, and are developed below.

3. Building Blocks: Recent Analytic Tools

Two new analytic tools developed in recent years are the basic ingredients of the procedure developed in this paper: IIS analysis for infeasible LPs, and nonviability analysis for network models.

3.1. IIS Analysis of Infeasible LPs

Van Loon[28] introduced the term *Irreducibly Inconsistent System (IIS)*, now also known as an *irreducible infeasible set*, to describe a set of constraints that is infeasible but that becomes feasible if any one constraint is removed. "Constraint" refers here to both rows and column bounds. Any infeasible LP contains at least one IIS.

The utility of the concept lies in the fact that the IIS is generally very much smaller than the original LP. This can provide a tremendous degree of localization of the infeasibility, in many cases as few as a couple of constraints from among the thousands defining the entire model. The IIS assists in the diagnosis of the problem by focusing attention on a few constraints, at least one of which must be changed if feasibility is to be achieved.

In a series of papers, Chinneck[7, 8] and Chinneck and Dravnieks[6] showed how to isolate IISs quickly and gave heuristics for finding small IISs where there are several in the model.[7] They further provided the first implementation of an IIS-finder by modifying a version of MINOS to create MINOS(IIS).[6, 8] The network analysis code PROFLOW, described in Section 4.2, uses MINOS(IIS) to solve the network models, and to analyze infeasibilities. Versions of Chinneck's IIS-finding algorithms have since been implemented in other codes, including CPLEX[9] and LINDO.[27] A related precursor algorithm also appears in the proprietary BP Oil code CLAUDIA.[24, 25]

For our purposes here, it is sufficient to know that IIS-isolation is practical and effective, and is available as an integral part of implemented LP software. For details of the method and for references to related work, see [6–8].

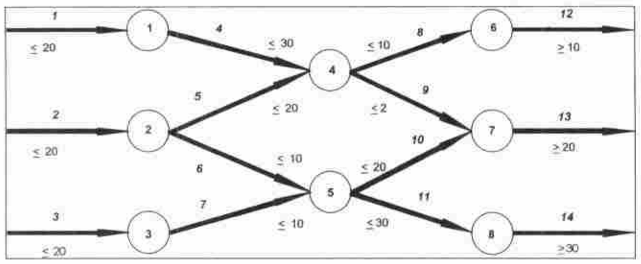

Figure 2. An infeasible pure network.[18, 20]

3.1.1. IIS Analysis of Network Models

A difference between flow balancing and IIS isolation methods is that the IIS also reports nonnegativity constraints that are implicated in the infeasibility, whereas flow balancing methods do not. Since a nonnegativity constraint in an IIS may indicate a reversed arc, this is important information. See Figure 2 for example, which has previously been analyzed by Greenberg[18] and Greenberg and Murphy.[20] Greenberg and Murphy identify the infeasibility as nodes 5 and 8, the upper bounds on arcs 6 and 7, and the lower bound on arc 14. The IIS lists the same constraints, but also includes the nonnegativity restriction on arc 10.

Many network representations have one redundant structural equation, normally an "environment" node showing the conservation of the flows exiting and entering the network itself. The redundant equation is often omitted implicitly by allowing nodes to act as sources or sinks. If the redundant structural equation is explicitly retained, then there are always two representations for each IIS: (i) an "internal" IIS which includes only nodes and arcs internal to the network, and (ii) an "external" IIS which includes the environment node. Experience shows that humans can more easily interpret the "internal" IIS, leading to the rule of thumb that if the redundant environment equation is included in the model, it is better to explicitly remove it before applying the IIS analysis to avoid finding the sometimes confusing "external" IIS.

IIS isolation has three main advantages over flow balancing for analyzing infeasible networks: (1) it provides a greater degree of localization of the infeasibility by guaranteeing the isolation of a single IIS, (2) involved nonnegativity constraints are explicitly identified so that reversed arcs are more easily identified, and (3) it is easily combined with other methods so that once an IIS is found, other standard methods, even supply and demand balancing, can be applied to this small portion of the original network, instead of to the entire model.

3.1.2. Aggregation of Network IISs

Network models sometimes have very large IISs, usually because incompatible input and output restrictions are linked via a large number of flow conservation equations. While this gives a lengthy list of constraints and flow bounds in the IIS, the infeasibility is conceptually simple: the upper limit on the input flows is less than the lower limit on the output flows, for example. Fortunately, aggregation of the IIS constraints can make this clear.

Aggregation refers to the summing of the regular node equations in the IIS to yield a simpler overall expression. This amounts to condensing the "bridge" of flow conservation equations linking the incompatible input and output flow restrictions, creating a single easily-understood equation. The basic idea of aggregation is not new,[26] but it has only recently been applied to assist in the formulation of LPs. For example, an AGGREGAT command is available in ANALYZE,[19] but the rows are weighted by the dual prices during the summation. Aggregating large network IISs by simple summing of the rows has been implemented by Ed Klotz in version 3.0 of the CPLEX optimizer.[9]

Consider the following small example as analyzed by CPLEX, which automatically provides the aggregation of the equality rows.

```
Rows in the IIS:
c125: − x50 + x379 − x380 = −1825
c126: − x379 + x380 − x382 = −2535
c127: − x381 + x382 + x383 − x384 = −1658
c128: − x30 − x383 + x384 + x387 − x459 = −15466
c147: − x69 + x435 − x437 = −338
c148: − x435 + x437 + x438 − x439 = −1037
c149: − x438 + x439 + x440 − x442 = −5713
c150: − x440 + x442 + x443 − x444 = −16
c151: − x443 + x444 + x446 − x448 = −1954
c153: − x446 + x448 + x449 − x450 = −4255
c154: − x449 + x450 + x451 − x453 = −5155
c155: − x451 + x453 + x454 − x455 = −1274
c156: − x454 + x455 + x456 + x457 − x458 − x463 = −1454
c157: − x387 − x456 + x458 + x459 = −6401
c158: − x457 + x463 + x464 − x491 = −14
c165: − x475 + x477 + x478 − x479 = −246
c166: − x478 + x479 + x480 − x482 = −232
c167: − x480 + x482 + x483 − x484 = −61
c168: − x483 + x484 + x485 − x486 = −1536
c169: − x485 + x486 + x487 − x488 = −3648
c170: − x487 + x488 + x489 − x490 = −3676
c171: − x464 − x489 + x490 + x491 = −1848

Column Bounds in the IIS:
x30 <= 12509
x50 <= 12509
x69 <= 14434
x475 <= 14434
x477 >= 0

Aggregated IIS Rows:
− x30 − x50 − x69 − x475 + x477 = −60342
```

The diagnosis of the error is difficult to make using only the listing of the rows and column bounds in the IIS, but is quite easily made by looking at the aggregated row and the column bounds.

Note that to be effective, simple summing of the regular nodes depends on using a consistent format for the regular node equations (e.g. outflows − inflows). Fortunately, reversals of the convention are easy to fix automatically.

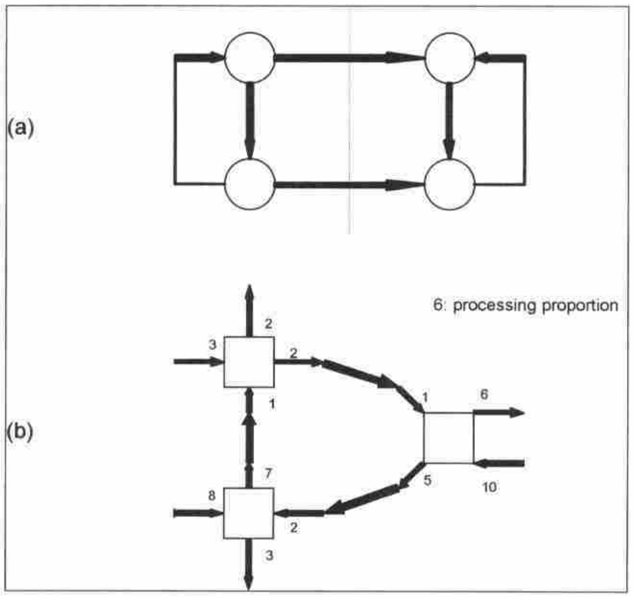

Figure 3. Simple examples of nonviable networks: (a) directed cutset, (b) full rank cycle of processing nodes.

3.2. Viability Analysis

Viability analysis[3, 5] is concerned with detecting and analyzing errors in the underlying structure of a network model that prevent one or more arcs from conducting flow. The network is first stripped of all nonstructural constraints (arc lower bounds are reset to zero, arc upper bounds are reset to infinity). If the structural constraints alone force one or more of the arc flows to zero, then the network is nonviable. Simple examples of nonviability include a directed cutset or a cycle of processing nodes of full rank (Figure 3).

Nonviability is usually an indication of a modeling error: why include arcs in the model if they can never conduct flow? Algorithms for localizing and diagnosing nonviability have been developed[3, 5] and implemented. One small-scale prototype implementation[4] is appropriate only for pure processing networks and uses a complex but more revealing algorithm. A second implementation in the PROFLOW software (Section 4.2) is suitable for very large problems and uses the IIS finding algorithms to localize a nonviability.

A nonviable network does not cause infeasibility, but a simple transformation converts a nonviability localization problem into an infeasibility localization problem that can be attacked using IIS methods.[5] The transformation works as follows. The feasible region of the stripped-down structural model investigated during viability analysis is a convex polyhedral cone. In a viable model, all variables can take on positive values simultaneously, but in a nonviable model one or more flow variables are restricted to zero. By adding a positivity constraint on every variable (generally a lower bound of 1), nonviability shows up as infeasibility, and IIS isolation methods can then be used to localize the cause of the nonviability.

The structural constraints examined during viability testing define the basic oriented interconnections of the network. This set of constraints is sometimes referred to as the "model," where "instances of the model" are created by adding strictly positive flow bounds and extra side constraints. In practice, the basic "model" often persists for long periods of time because the physical infrastructure of the modeled system does not change, whereas nonstructural constraints such as flow bounds and side constraints may change often, reflecting new conditions of output demands, input availabilities, etc.

4. A Procedure for Analyzing Infeasible Networks

The recommended procedure for analyzing infeasible networks uses an ordered sequence of tests to extract extra diagnostic information. The diagnosis is improved by establishing confidence in the model at various levels, moving from the most fundamental (the underlying structure of the node interconnections and arc orientations), to considerations of individual node and arc capacity interactions (often related to the sizes of physical equipment), to overall input and output constraints. The ordered set of tests is given in Algorithm 1. There are two important orderings in the set of tests, as explained next.

Algorithm 1. The Sequence of Tests.

1. Assemble only the structural equations and apply the viability test. If the network is nonviable, then localize the nonviability and exit.
2. Add the simple arc flow bounds and test feasibility. If the network is infeasible, then localize the IIS and exit.
3. Add the extra side constraints and test feasibility. If the network is infeasible, then localize the IIS and exit.

The first ordering requires that viability testing be carried out before feasibility testing because nonviability has special meaning when analyzing infeasible networks. Specifically, if a network is nonviable and a strictly positive lower flow bound is imposed on a nonviable arc, then the model will be infeasible. Further, any attempts to repair the model by adjusting the lower flow bound to some other strictly positive value cannot succeed. The viability error in the underlying network structure must be repaired before model instances are examined.

The second ordering examines the structural constraints plus simple flow bounds (step 2) before the entire model, which includes the extra side constraints (step 3). Simple flow bounds are usually limiting physical factors such as pipe diameter which often persist in the model for lengthy periods, like the underlying structure. It is therefore useful to check that these are not the cause of infeasibilities in conjunction with the structure before examining the effect of any other constraints. Extra side constraints are often some form of aggregate input or output requirement. These are most likely to change in creating a model instance.

In addition to localizing the error, the ordering of the tests in Algorithm 1 provides basic diagnostic information on the *cause* of the error: structure, equipment sizing, or input/output requirements.

4.1. Algorithm Efficiency

The IIS finding routines that drive all three steps of Algorithm 1 are remarkably fast. Many versions of the algorithms are available, with the fastest having an average time ratio (time to find IIS: time to complete initial phase 1 signaling infeasibility) of 0.9, and the recommended method that gives the most useful IISs having an average time ratio of 3.4.[7] The time taken to complete all three steps of the algorithm (if necessary), is considerably less than three times the time taken to complete one step because the final solution of one step can be used as an advanced start for the next step. Such advanced start capabilities are available in MINOS(IIS), which is embedded in PROFLOW and in other modern solvers.

At the (extremely unlikely) worst, with all three algorithm steps experiencing a time ratio of 3.4, the time ratio for the entire algorithm would be approximately 10. In other words, at worst it might take at most about 10 times as long to run the complete algorithm as to detect phase 1 infeasibility in the first place, but in most cases the time ratio will be much smaller. These are very favourable figures, especially considering that the real issue is minimizing *human time* for model debugging. In practice, managers are glad to expend machine time to reduce human time. For a relatively small expenditure of machine time, the algorithm provides an isolation and a preliminary diagnosis of the error.

IIS localization algorithms have not yet been incorporated in specialized codes for solving network models (which can be 10–200 times faster than the general simplex method,[1, 2, 14, 16] but this should be straightforward. The speed advantage of a specialized network code depends heavily on the number of side constraints (such as processing proportion equations) and extra side constraints. Using a specialized network code in place of a general LP code such as MINOS(IIS) will also speed the analysis process.

4.2. The PROFLOW Software

PROFLOW is a software tool that has been developed by the author for formulating, analyzing, and solving network models of all types, and that incorporates most of the elements of the algorithm described above. The main features are (1) an intuitive language for describing network models of all types, including extra side constraints, (2) automatic setup and solution of the minimum cost network flow optimization, (3) automatic localization of nonviabilities on request, (4) automatic IIS localization of infeasibilities, (5) a friendly interface including windowing, pull-down menus, context-sensitive help, a full-featured editor, easy browsing of solutions, and automatic checking of the network description for syntax errors and basic network errors such as unconnected arcs. An example of the PROFLOW network description language can be seen in Figure 5.

MINOS(IIS) is the embedded solver and provides the IIS localization when needed for either nonviability or infeasibility analysis. At present, the ordered set of tests described in Algorithm 1 must be set up manually, but it is intended that this will be automated in future.

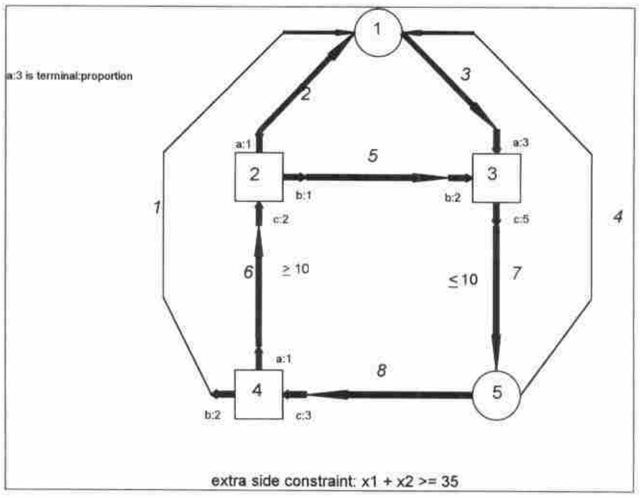
a:3 is terminal:proportion
1
2
3
4
5
6
7
8
a:1
a:3
b:1
b:2
c:2
c:5
≥ 10
≤10
a:1
b:2
c:3
extra side constraint: x1 + x2 >= 35

Figure 4. Small example network.

5. Examples

An initial small illustrative example is given in Figure 4. To understand the error-localizing power of the algorithms, imagine that the small network of Figure 4 is attached to a very large network via more arcs incident on node 1. The self-explanatory PROFLOW language description of the network is given in Figure 5.

The state-of-the-art flow balancing methods cannot be applied in this example because it is not an ordinary pure network. Attempting to apply flow balancing by relaxing the proportion constraints on the processing nodes and by ignoring the extra side constraints leads to the incorrect conclusion that the network is feasible (i.e., a flow of 10 units in arcs 5, 6, 7, 8).

Using PROFLOW to apply the general LP IIS isolation algorithm leads to the IIS listed in Figure 6, which is the portion of the network shown in Figure 7. This is indeed a modeling error that needs repair, but it is not the only problem, as shown below.

Analysis by the recommended procedure turns up additional errors needing repair. The viability analysis in step 1 reveals the nonviability listed in Figure 8, which is the portion of the network shown in Figure 9. No amount of positive flow can be sustained on the arcs in Figure 9 until the nonviability is repaired, in this case by correcting the accidental reversal of arc 4. When the orientation of arc 4 is corrected, step 1 now concludes that the model is viable.

Step 2 adds the flow bounds and tests the feasibility of the model. This reveals the same IIS identified in Figures 6 and 7. The infeasibility is due to the incorrect assignment of an upper flow bound of 10 to arc 7, which limits the output of the process represented by node 3. When the flow bound on arc 7 is correctly set to 15, the next application of Step 2 shows the model is now flow-bound feasible.

Step 3 adds the extra side constraint and tests the feasibility of the whole model. The IIS listed in Figure 10 and pictured in Figure 11 is revealed. Knowing that the model is flow-bound-feasible, it is natural to focus on the extra side

```
DEFNETWORK paper_eg_1:

  REGULAR: node1, node5;

  DEFPROCESSOR second_node:
    LABEL=node2;
    DEFTERMINAL top_term:
      LABEL=terma; RATIO=1.0; ENDTERMINAL;
    DEFTERMINAL side_term:
      LABEL=termb; RATIO=1.0; ENDTERMINAL;
    DEFTERMINAL bot_term:
      LABEL=termc; RATIO=2.0; ENDTERMINAL;
  ENDPROCESSOR;

  DEFPROCESSOR third_node:
    LABEL=node3;
    DEFTERMINAL top_term:
      LABEL=terma; RATIO=3.0; ENDTERMINAL;
    DEFTERMINAL side_term:
      LABEL=termb; RATIO=2.0; ENDTERMINAL;
    DEFTERMINAL bot_term:
      LABEL=termc; RATIO=5.0; ENDTERMINAL;
  ENDPROCESSOR;

  DEFPROCESSOR fourth_node:
    LABEL=node4;
    DEFTERMINAL top_term:
      LABEL=terma; RATIO=1.0; ENDTERMINAL;
    DEFTERMINAL left_term:
      LABEL=termb; RATIO=2.0; ENDTERMINAL;
    DEFTERMINAL right_term:
      LABEL=termc; RATIO=3.0; ENDTERMINAL;
  ENDPROCESSOR;

  DEFEDGE first_arc: LABEL=arc1; COST=-5.0;
    FROMNODE=(node4,termb); TONODE=(node1); ENDEDGE;
  DEFEDGE second_arc: LABEL=arc2; COST=-5.0;
    FROMNODE=(node2,terma); TONODE=(node1); ENDEDGE;
  DEFEDGE third_arc: LABEL=arc3; COST=2.5;
    FROMNODE=(node1); TONODE=(node3,terma); ENDEDGE;
  DEFEDGE fourth_arc: LABEL=arc4; COST=2.0;           (reversed!)
    FROMNODE=(node5); TONODE=(node1); ENDEDGE;
  DEFEDGE fifth_arc: LABEL=arc5;
    FROMNODE=(node2,termb); TONODE=(node3,termb); ENDEDGE;
  DEFEDGE sixth_arc: LABEL=arc6; MINFLOW=10.0;
    FROMNODE=(node4,terma); TONODE=(node2,termc); ENDEDGE;
  DEFEDGE seventh_arc: LABEL=arc7; MAXFLOW=10.0;
    FROMNODE=(node3,termc); TONODE=(node5); ENDEDGE;
  DEFEDGE eighth_arc: LABEL=arc8;
    FROMNODE=(node5); TONODE=(node4,termc); ENDEDGE;

  SIDECONS extra_side: arc1 + arc2 > 35.0;

ENDNETWORK.
```

Figure 5. PROFLOW language description of example 1.

```
IIS generated by infeasibility:

--------------- Columns ---------------

ARC6                          lower bound
ARC7                          upper bound

---------------- Rows -----------------

NODE2[TERMC:TERMB]            fixed value
NODE3[TERMC:TERMB]            fixed value
```

Figure 6. PROFLOW output identifying an IIS.

constraint, determining in this case that the lower bound of 35 on the total output of the network is too high.

This example shows that flow balancing is ineffective for network forms beyond pure ordinary networks. In addition, the general LP IIS isolation applied directly to the model provides only part of the diagnosis: the underlying nonviable structure should be corrected first, since *any* strictly positive lower bound on a nonviable arc will cause infeasibility.

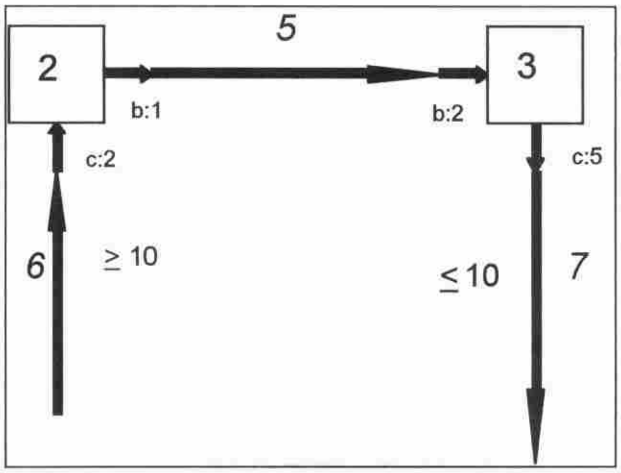

Figure 7. Network portion isolated by the IIS.

```
IIS generated by nonviability:

--------------- Columns ---------------

ARC4                          lower bound
ARC6                          lower bound

---------------- Rows -----------------

NODE2[TERMC:TERMB]            fixed value
NODE3[TERMC:TERMB]            fixed value
NODE4[TERMC:TERMA]            fixed value
NODE5                         fixed value
```

Figure 8. Step 1 nonviability isolated by PROFLOW.

Since they are too large to show graphically, a second and third example are summarized below. Processing network model MODEL61I is a multiperiod forest growth and harvest model consisting of 190 nodes (100 processing, 88 regular, 2 external), 367 arcs, and 6 side constraints. Algorithm step 1 shows that the model is viable, but step 2 finds an IIS consisting of 8 arcs and 13 nodes. This is a small portion of the total model (arcs: $8/367 = 0.02$, nodes: $13/190 = 0.07$), and allows swift identification and correction of an incorrectly set arc lower bound. Algorithm step 3 isolates an IIS involving 2 side constraints, 11 arcs, and no nodes. This is again a small portion of the model (arcs: $11/367 = 0.03$, nodes: $0/190 = 0.0$, side constraints: $2/6 = 0.3$).

Processing network model EX73AI was generated as part of a method for analyzing software designs by a processing network transformation,[29] and consists of 158 nodes (88 processing, 70 regular) and 211 arcs. Algorithm step 1 shows that the model is nonviable, and the isolated nonviability consists of 25 arcs and 21 nodes, an effective isolation (arcs: $25/211 = 0.12$, nodes: $21/158 = 0.13$). When the nonviability is corrected, a small infeasibility (2 arc bounds, one regular node) is detected and corrected. There are no side constraints.

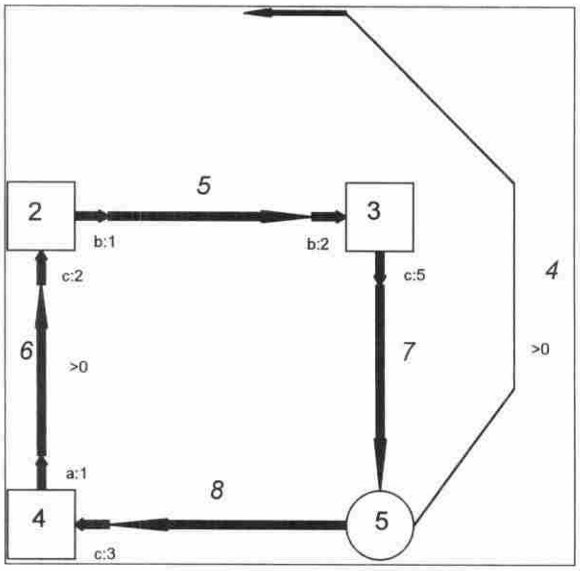

Figure 9. Portion of network isolated by the nonviability.

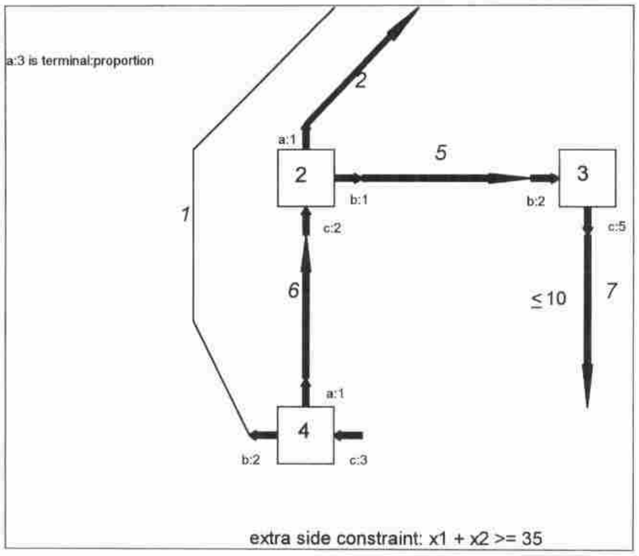
a:3 is terminal:proportion
2
a:1
5
2
3
1
b:1
b:2
c:2
c:5
6
≤10
7
a:1
4
b:2
c:3
extra side constraint: x1 + x2 >= 35

Figure 11. Network portion isolated by the Step 3 IIS.

```
IIS generated by infeasibility:

--------------- Columns ---------------

ARC7                                upper bound

---------------- Rows -----------------

NODE2[TERMC:TERMB]                  fixed value
NODE2[TERMC:TERMA]                  fixed value
NODE3[TERMC:TERMB]                  fixed value
NODE4[TERMC:TERMB]                  fixed value
NODE4[TERMC:TERMA]                  fixed value
EXTRA_SIDE                          lower bound
```

Figure 10. Step 3 IIS isolated by PROFLOW.

6. Conclusions

The procedure for the analysis of infeasible network models presented here provides an improved analysis for several reasons. First, it applies to all network forms. Flow balancing methods, the current state of the art, apply only to pure ordinary networks and may provide insufficient information to give a useful localization or to form a complete diagnosis even for this class of models.

Second, the general LP IIS isolation applied directly to the network model does not extract all of the possible diagnostic information. By including nonviability analysis and by examining various classes of constraints in an ordered series of tests, the new procedure provides added diagnostic value. It first determines whether the basic structure is workable (i.e., viable), then whether the flow capacities (e.g., equipment capacities) are feasible, and finally whether the side constraints (e.g., total input and output limits) are feasible.

Finally, the suggested procedure is easily implemented, making use of known building blocks which are easily incorporated into standard network modeling systems. A prototype implementation, PROFLOW, already exists.

Acknowledgments

The support of this research by the Natural Sciences and Engineering Research Council of Canada through a research grant to the author is gratefully acknowledged. Marco Lavoie did an excellent job of programming the PROFLOW interface. Thanks also to Harvey J. Greenberg (University of Colorado at Denver) for advice during the preparation of early drafts of this paper.

References

1. C-H.J. CHEN and M. ENGQUIST, 1986. A Primal Simplex Approach to Pure Processing Networks, *Management Science 32*, 1582–1598.
2. S. CHEN and R. SAIGAL, 1977. A Primal Algorithm for Solving a Capacitated Network Flow Problem with Additional Linear Constraints, *Networks 7*, 59–79.
3. J.W. CHINNECK, 1990. Formulating Processing Network Models: Viability Theory, *Naval Research Logistics 37*, 245–261.
4. J.W. CHINNECK, 1990. VIABLE1: Code for Identifying Nonviabilities in Processing Network Models, technical note in the Operations Research Software Exchange Program column, *European Journal of Operational Research 44*, 119.
5. J.W. CHINNECK, 1992. Viability Analysis: A Formulation Aid For All Classes of Network Models, *Naval Research Logistics 39*, 531–543.
6. J.W. CHINNECK and E.W. DRAVNIEKS, 1991. Locating Minimal Infeasible Constraint Sets in Linear Programs, *ORSA Journal on Computing 3*, 157–168.
7. J.W. CHINNECK, 1993. Finding the Most Useful Subset of Constraints for Analysis in an Infeasible Linear Program, Technical report SCE-93-07, Systems and Computer Engineering, Carleton University, Ottawa, Canada.
8. J.W. CHINNECK, 1994. MINOS(IIS): Infeasibility Analysis Using MINOS, *Computers & Operations Research 21*, 1–9.

9. CPLEX OPTIMIZATION INC., 1994. *Using the CPLEX Callable Library and CPLEX Mixed Integer Library*, version 2.2, software manual.
10. L.R. FORD and D.R. FULKERSON, 1962. *Flows in Networks*, Princeton University Press, Princeton, NJ.
11. D.R. FULKERSON, 1959. A Network Flow Feasibility Theorem and Combinatorial Applications, *Canadian Journal of Mathematics 11*, 440–451.
12. D. GALE, 1957. A Theorem in Networks, *Pacific Journal of Mathematics 7*, 1073–1082.
13. J. GLEESON and J. RYAN, 1990. Identifying Minimally Infeasible Subsystems of Inequalities, *ORSA Journal on Computing 2*, 61–63.
14. F. GLOVER and D. KLINGMAN, 1981. The Simplex SON Algorithm for LP/Embedded Network Problems, *Mathematical Programming Study 15*, 148–176.
15. F. GLOVER, D. KLINGMAN and N.V. PHILLIPS, 1992. *Network Models in Optimization and Their Applications in Practice*, John Wiley & Sons Inc., New York.
16. H.J. GREENBERG, 1987. Computer-Assisted Analysis for Diagnosing Infeasible or Unbounded Linear Programs, *Mathematical Programming Studies 31*, 79–97.
17. H.J. GREENBERG, 1987. Diagnosing Infeasibility for Min-Cost Network Flow Models, Part I: Dual Infeasibility, *IMA Journal of Mathematics in Management 1*, 99–109.
18. H.J. GREENBERG, 1988. Diagnosing Infeasibility for Min-Cost Network Flow Models, Part II: Primal Infeasibility, *IMA Journal of Mathematics Applied in Business and Industry 2*, 1–12.
19. H.J. GREENBERG, 1993. *A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE*, Kluwer Academic Publishers, Boston.
20. H.J. GREENBERG and F.H. MURPHY, 1991. Approaches to Diagnosing Infeasible Linear Programs, *ORSA Journal on Computing 3*, 253–261.
21. J. HAO and J.B. ORLIN, 1992. Diagnosing Infeasibilities for Minimum Cost Flow Problems, TIMS/ORSA Conference, Orlando Florida, April 26–29.
22. A.J. HOFFMAN, 1960. Some Recent Applications of the Theory of Linear Inequalities to Extremal Combinatorial Analysis, *Proceedings of Symposia on Applied Mathematics 10*.
23. J. KOENE, 1982. *Minimal Cost Flow in Processing Networks, a Primal Approach*, Ph.D. thesis, Eindhoven University of Technology, Eindhoven, The Netherlands. Also released as CWI Tract 4, 1984.
24. R.A. MAIN, 1993. Infeasibility Analysis Using CLAUDIA-I, Technical report, BP Oil International.
25. R.A. MAIN, 1993. Infeasibility Analysis Using CLAUDIA-II, Technical report, BP Oil International.
26. K.G. MURTY, 1983. *Linear Programming*, John Wiley & Sons, New York.
27. L. SCHRAGE, 1991. *LINDO: An Optimization Modelling System*, 4th ed., The Scientific Press, San Francisco.
28. J. VAN LOON, 1981. Irreducibly Inconsistent Systems of Linear Inequalities, *European Journal of Operational Research 8*, 283–288.
29. Z. YOU, 1993. *Localization and Diagnosis of Structural Problems in Petri Net Models*, M.Sc. thesis, Systems and Computer Engineering, Carleton University, Ottawa, Canada.