Analyzing Infeasible Optimization Models

John W. Chinneck
Systems & Computer Engineering
Carleton University
Ottawa, Canada

A tutorial for CORS/INFORMS 2004
May 16-19, Banff, Canada

Why is (In)feasibility Interesting?

- Sometimes **any** feasible solution will do.
- Feasibility question can be same as **optimality** question.
- Assistance in formulating complex optimization models: **why** is it infeasible?
- **Applications** of infeasibility analysis:
  - Training neural networks
  - Classification via math programming methods
  - Radiation treatment planning
  - Backtracking in constraint logic programs
  - Applications to NP-hard problems
  - Statistical analysis
  - Protein folding …

Outline

1. Analyzing Infeasible Math Programs
    1. Infeasibility Isolation
        1. General Methods
        2. Linear Programming
        3. Mixed-Integer Programming
        4. Nonlinear Programming
    2. Finding Maximum Feasible Subsets
    3. Software
    4. Applications:
        1. Formulation: Networks; Multi-Objective Programs, etc.
        2. Other Applications
2. Faster Feasibility
    1. Mixed-Integer Programs
    2. Nonlinear Programs

1. Analyzing Infeasible Math Programs

Two main approaches:

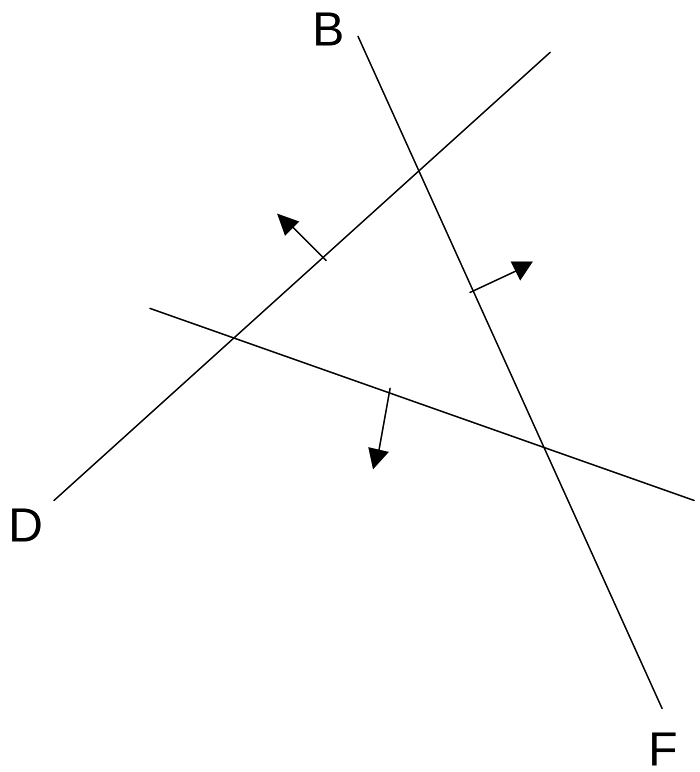
- Isolate an ***Irreducible Infeasible System***
  - An infeasible set of constraints that becomes feasible if any constraint removed
- Find a ***Maximum Feasible Subset***
  - Maximum cardinality subset of constraints that is feasible

1.1 Infeasibility Isolation

Using IISs

Cycle:

1. Isolate an IIS
2. Repair the infeasibility
3. If still not feasible, go to step 1.



1.1.1 General Methods for Finding IISs

- Suppose the solver is perfectly accurate in deciding feasibility status of a set of constraints
- General methods for IIS isolation:
  - Deletion Filter
  - Additive Method
  - Elastic Filter
  - Additive/Deletion method

The Deletion Filter

INPUT: an infeasible set of constraints.

FOR each constraint in the set:

- Temporarily drop the constraint from the set.
- Test the feasibility of the reduced set:
- IF feasible THEN return dropped constraint to the set.
- ELSE (infeasible) drop the constraint permanently.

OUTPUT: constraints constituting a single IIS.

Deletion Filter: Example

IIS is {B,D,F} in {A,B,C,D,E,F,G}

- {B,C,D,E,F,G} infeasible. A deleted.
- {C,D,E,F,G} feasible. B reinstated.
- {B,D,E,F,G} infeasible. C deleted.
- {B,E,F,G} feasible. D reinstated.
- {B,D,F,G} infeasible. E deleted.
- {B,D,G} feasible. F reinstated.
- {B,D,F} infeasible. G deleted.

Output: the IIS {B,D,F}

Deletion Filter: Characteristics

- Returns *exactly one* IIS, even if there are multiple IISs in the model
- Which IIS?
  - IIS whose *first* member is *last* in the test list.
- Speed: isn't this slow?
  - time to isolate IIS is normally a small fraction of time to find infeasibility initially
  - Due to advanced starts: each LP is very similar to the previous one

The Additive Method

C: ordered set of constraints in the infeasible model.
T: the current test set of constraints.
I: the set of IIS members identified so far.

INPUT: an infeasible set of constraints C.
Step 0: Set $T = I = \varnothing$.
Step 1: Set $T = I$.
FOR each constraint c_i in C:
Set $T = T \cup c_i$.
IF T infeasible THEN
Set $I = I \cup c_i$.
Go to Step 2.
Step 2: IF I feasible THEN go to Step 1.
OUTPUT: I is an IIS.

Additive Method: Example

IIS is {B,D,F} in {A,B,C,D,E,F,G}

- {A}, {A,B}, {A,B,C}, {A,B,C,D}, {A,B,C,D,E} all feasible.
- {A,B,C,D,E,F} infeasible: *I* = {F} is feasible.
- {F,A}, {F,A,B}, {F,A,B,C} all feasible.
- {F,A,B,C,D} infeasible: *I* = {F,D} is feasible.
- {F,D,A} feasible.
- {F,D,A,B} infeasible: *I* = {F,D,B} infeasible. Stop.

Output: the IIS {F,B,D}

Additive Method: Characteristics

- Returns *exactly one* IIS, even if there are multiple IISs in the model
- Which IIS?
  - IIS whose *last* member is *first* in the test list.
- Speed:
  - Similar to deletion filter due to basis re-use
  - If IIS is small and early in the list of constraints, can use far fewer LP solutions that deletion filter

Speed-up: Grouping Constraints

- Add/drop constraints in groups
  - In order, or by category
- *Elastic Filter*: back up and do singly if dropping a group causes feasibility
- *Additive Method*: back up and do singly if adding a group causes infeasibility

- Can speed up the methods
  - Fix group size? Adaptive group sizing?

Additive/Deletion Method

1. Apply additive method until first infeasible subset of constraints is found.
2. Apply deletion filter to subset.

- More efficient.

Elasticizing Constraints

- Make all constraints elastic by adding elastic variables, e_i
- Elastic objective: Min Σe_i

Original constraint	elastic version
$g(x) \geq b_i$	$g(x) + e_i \geq b_i$
$g(x) \leq b_i$	$g(x) - e_i \leq b_i$
$g(x) = b_i$	$g(x) + e_i' - e_i'' = b_i$

The Elastic Filter

INPUT: an infeasible set of constraints.

1. Make all constraints elastic by incorporating nonnegative elastic variables e_i.
2. Solve the model using the elastic objective function.
3. IF feasible THEN

   Enforce the constraints in which any e_i>0 by permanently removing their elastic variable(s).

   Go to step 2.

   ELSE (infeasible): Exit.

OUTPUT: the set of de-elasticized constraints contains at least one IIS.

The Elastic Filter: Example

IIS is {B,D,F} in {A,B,C,D,E,F,G}

Elasticized constraints are underscored.

- {$\underline{\text{A}}$,$\underline{\text{B}}$,$\underline{\text{C}}$,$\underline{\text{D}}$,$\underline{\text{E}}$,$\underline{\text{F}}$,$\underline{\text{G}}$} feasible. B stretched.
- {$\underline{\text{A}}$,B,$\underline{\text{C}}$,$\underline{\text{D}}$,$\underline{\text{E}}$,$\underline{\text{F}}$,$\underline{\text{G}}$} feasible. F stretched.
- {$\underline{\text{A}}$,B,$\underline{\text{C}}$,$\underline{\text{D}}$,$\underline{\text{E}}$,F,$\underline{\text{G}}$} feasible. D stretched.
- {$\underline{\text{A}}$,B,$\underline{\text{C}}$,D,$\underline{\text{E}}$,F,$\underline{\text{G}}$} infeasible.

Output: the set {B,F,D}

  - Not necessarily an IIS until deletion filtered

The Elastic Filter: Characteristics

- At least one member of *every* IIS will stretch at each iteration
- Number of iterations: at most equal to cardinality of *smallest* IIS
  - Useful in finding small IISs
- Output needs deletion filter to identify a single IIS

1.1.2 Special Methods for LP

Bound-Tightening

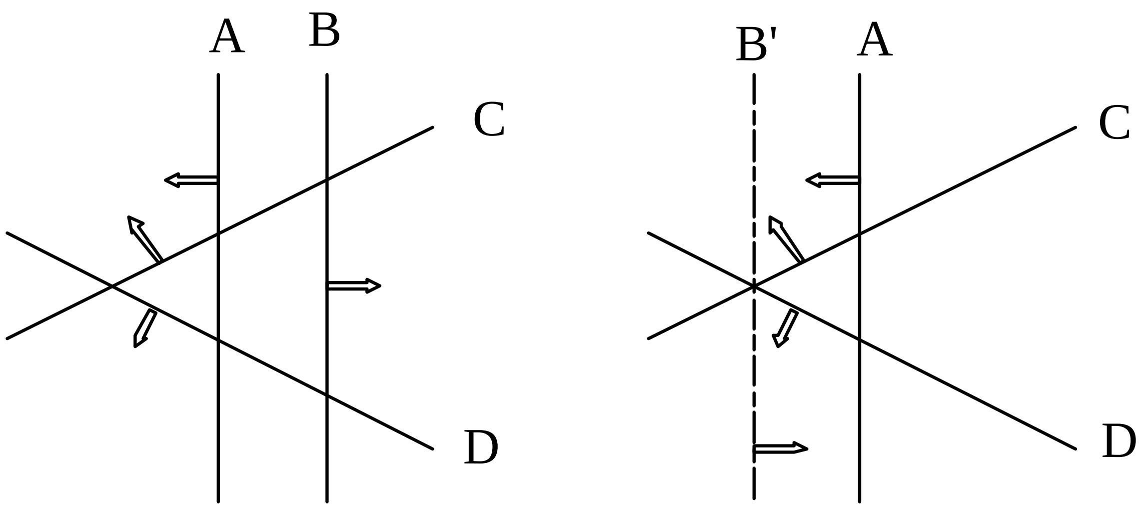
- Standard presolver techniques: iterative tightening of bounds. E.g.:
  - $2x_1 - 5x_2 \leq 10$ where $-10 \leq x_1, x_2 \leq 10$
  - Apply constraint with x_1 is at it's lower bound: $2(-10) - 5x_2 \leq 10 \Rightarrow x_2 \geq -6$.
  - Lower bound on x_2 tightened.
- May lead to detection of infeasibility.
- Difficult to deduce IIS from long sequence of operations.

The Sensitivity Filter

- Drop all constraints to which the phase 1 objective is not sensitive
  - Insensitive if dual variable is zero
  - Can apply when infeasibility first detected
- Characteristics:
  - Eliminates many constraints very quickly
  - Tends to lead to larger IISs

Sensitivity Filter: Characteristics

- Tends to isolate larger IISs



a) before: two IISs, {A,B} and {B,C,D}.

b) after: one IIS, {B',C,D}.

Constraints shift during phase 1

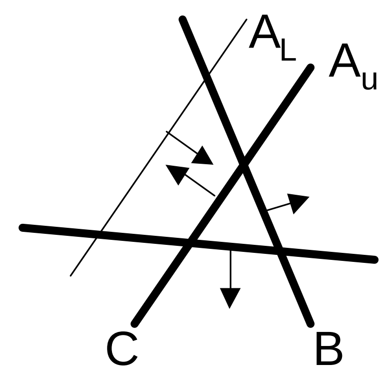
Interior Point Methods

- Solution from interior point method can separate the set of constraints into two parts:
  - those that might be part of **some** IIS
  - those that are irrelevant to **any** IIS.
- Theorem on strictly complementary partitions.
- Some advantages over the sensitivity filter, which cannot always identify *all* the constraints that are part of *some* IIS

Deletion/Sensitivity, Reciprocal Filters

Deletion/Sensitivity Filter

- Apply sensitivity filter each time deletion filter deletes a constraint permanently



Reciprocal Filter

- For ranged constraints
- Barring simple bound reversal:
  - If one of the bounds is involved in an IIS, then the other bound cannot be in the same IIS

Simplex Pivoting

- ***A:*** *$p \times n$* matrix (nonnegativity constraints included in $\boldsymbol{Ax} \leq \boldsymbol{b}$),
- *Theorem:* $\boldsymbol{Ax} \leq \boldsymbol{b}$, $\boldsymbol{x},\boldsymbol{b} \geq \boldsymbol{0}$, is an IIS iff:
  - there exist *(p-1)* linearly independent rows, and
  - there exist $\lambda > \mathbf{0}$ such that $\Sigma\lambda_i a_{ij} = 0$ and $\Sigma\lambda_i b_i < 0$.
- Efficient pivoting schemes to find such systems

Simplex Pivoting: Characteristics

- Problem size blows up when equalities converted
- Generally slower than filtering methods
- Not commercially implemented

Guiding the IIS Search

- Mark some constraints prior to IIS search:
  - remove immediately
  - encourage removal
  - discourage removal
  - never remove
- Give constraints different weights during elastic filter
- Why might this be done?
  - It is known that parts of the model are OK
  - There are several “reflections” of the same IIS, some easier to understand than others.
- Available in MINOS(IIS) [1994] and Cplex 9.0 [2003].

Finding Better IISs in LPs

- Model may have multiple IISs representing the *same* infeasibility
- IISs having few row constraints preferred
- General rules:
  - Avoid the sensitivity filter
  - Deletion filter: test row constraints before column bounds
  - Retain the column bounds for as long as possible to permit more rows to be eliminated during filtering
  - Use elastic filtering on the row constraints.
- Most effective heuristic tested:
  1. elastic filter the row constraints
  2. deletion/sensitivity filter the row constraints while protecting the variable bounds
  3. sensitivity filter the variable bounds
  4. deletion/sensitivity filter the variable bounds

Networks: Supply-Demand Balancing

- Logical reductions based on supply and demand connected via balance nodes
  - Uses theorems by Gale, Fulkerson, Hoffman
  - Hao and Orlin: use maximum flow algorithm to find a minimal "witness" set of nodes for which the net supply and the total outflow capacity conflict.
- Similar to presolver bound reductions
- Difficult to arrive at solid diagnosis by following the sequence of reductions
- Methods work only on simple network forms.

Networks: Aggregating Large IISs

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

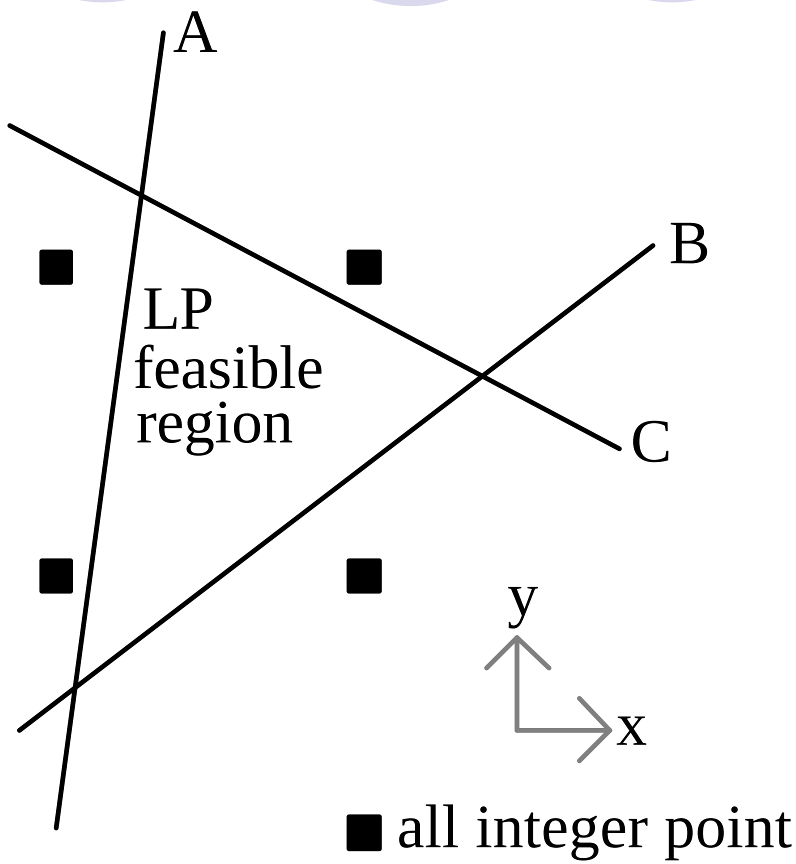
Aggregate sum of the balance constraints:

- x30 - x50 - x69 - x475 + x477 = -60342

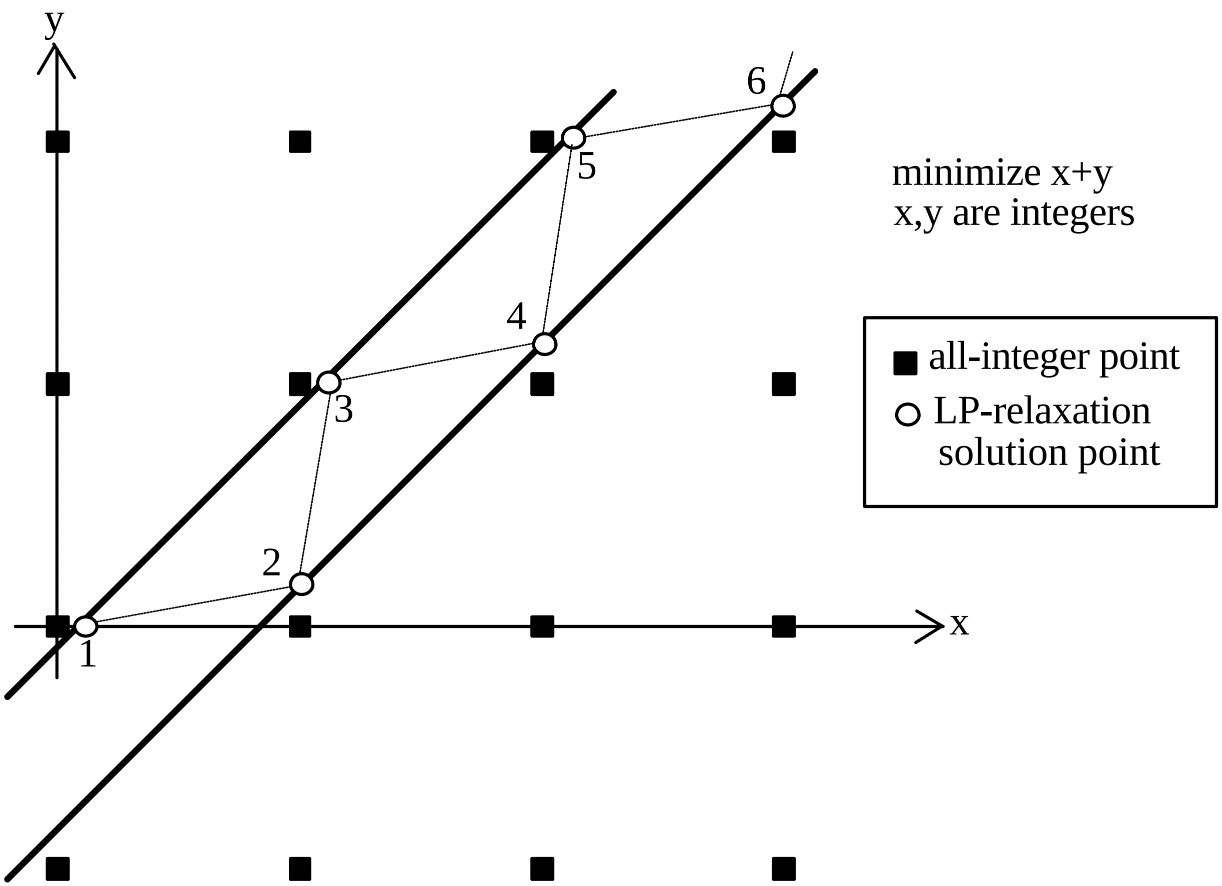
Before: 22 rows, 5 bounds, numerous variables
After: 1 row, 5 bounds, 5 variables

1.1.3 Special Methods for MIPs

- Three classes of constraints:
  - Linear row constraints (LC)
  - Variable bounds (BD)
  - *Integer Restrictions (IR)*



Nontermination in MIPs



Simple Deletion Filtering for MIPs

- Test rows, bounds, and integer restrictions
- Can suffer from nontermination
  - Test variable bounds *last*
  - If computation limit exceeded on subproblem, retain constraint and label it *dubious*
  - Get “infeasible subsystem” (IS) instead of IIS if there are dubious constraints
- Very slow
  - Each test requires full B&B tree expansion
  - Test integer restrictions *first:* IR-LC-BD method
  - Often returns small IS instead of IIS

Additive Method for MIPs

- Assume initial LP is feasible
  - Add IRs to LC∪BD
- Cannot identify dubious constraints
- Dynamic Reordering variant:
  - When a subproblem is feasible:
    - scan all constraints later in list; add all constraints satisfied at current solution point to *T*
- Additive/Deletion Method
  - Identifies dubious constraints via deletion filter

Using the Initial B&B Tree

- What can the initial B&B tree that detected infeasibility tell us?

1. No IIS has IR set identical to the set of IRs satisfied at any intermediate node.
2. Mark sensitive LCs and BDs at all leaf nodes. IR∪{marked LCs}∪{marked BDs} is infeasible.
   - Some LCs and BDs can be eliminated
3. *LC*∪*BD*∪{IRs on all branching variables} is infeasible.
   - IRs not in this set can be eliminated
   - Get candidate ISs by looking at sets of IRs defined by root-to-leaf paths.

Speed-ups for MIPs

- Grouping constraints for additive method, deletion filter
  - Numerous schemes, including adaptive sizing of groups
- Safety Bounds
  - Extra BDs to prevent nontermination
  - If triggered, then output is an IS

Testing MIP Algorithms

- 20 MIPLIB models altered to be infeasible
- Average time for initial detection of infeasibility: 0:0:6 (h/m/s)
- Average time for infeasibility analysis:
  - simple LC-IR-BD deletion filter: *9:12:46* (few dubious)
  - simple IR-LC-BD deletion filter: *2:27:44* (few dubious)
  - IR-LC-BD deletion, groups of 4: *1:51:31* (few dubious)
  - simple additive method: *1:12:12* (3 killed)
  - dyn. reordering additive method: *0:19:41* (2 killed)
  - dyn. reorder. add./del. method: *2:25:21* (fewest dubious)

1.1.4 Special Methods for NLP

- NLP solvers are not perfectly accurate in deciding feasibility.
  - *Factors:* NLP algorithm and implementation, tolerances, initial point, termination criteria, method of approximating derivatives, etc.
  - If feasibility detected: status is certain
    if unable to find feasible pt.: status is unknown
- **Minimal Intractable Subsystem (MIS):** minimal set of constraints causing NLP solver to report infeasibility with a given set of parameter settings (including initial point, tolerances, termination conditions, etc.)
- Missing constraints can cause math errors: sqrt(x), x = 0
  - *Guard constraints* prevent math errors

Deletion Filter for NLPs

INPUT: an infeasible set of nonlinear constraints.
FOR each constraint in the set:

1. reset the initial point and solver parameters.
2. temporarily drop the constraint from the set.
3. test the feasibility of the reduced set and DO CASE:
    - i. solver reports feasibility:
        - return dropped constraint to the set.
    - ii. solver reports infeasibility (ordinary):
        - drop constraint permanently.
    - iii. solver reports infeasibility (math error):
        - a. mark dropped constraint as a guard.
        - b. return dropped constraint to the set.

OUTPUT: constraints constituting a single MIS (including guards).

Four Possible Outcomes

- Model is feasible:
  - correctly detected by solver. No analysis. *(best)*
  - reported infeasible by solver and MIS isolated. *(worst)*
- Model is infeasible:
  - MIS is isolated which is also an IIS. *(best)*
  - MIS is isolated which is not an IIS. *(acceptable)*
- Worst case interpretation: <u>*this*</u> solver finds <u>*this*</u> MIS intractable with <u>*these*</u> setting

IIS Isolation: State of the Art

- LP: *mature*
  - well developed
  - commercially implemented
- MIP: *research opportunities*
  - needs faster methods,
  - needs improved ability to find IISs vs. ISs
- NLP: *research opportunities*
  - needs more reliable methods
  - needs improved accuracy in deciding feasibility

1.2 Finding Maximum Feasible Subsets

- Equivalent Problems on an infeasible set of linear constraints:
  - **`MAX FS`**: find max cardinality feasible subset
  - **`MIN ULR`**: find min cardinality subset of constraints to remove so that remaining set is feasible
  - **`MIN IIS COVER`**: find smallest cardinality subset of constraints to remove such that at least one constraint is removed from every IIS
- Problem is NP-hard
  - Are there good heuristics?
- MIN IIS COVER is not unique

Method of Parker and Ryan

- Use a simplex pivoting method to generate IISs one at time
- As each IIS is generated, add it to the set of known IISs, then solve a set-covering problem via integer programming
- Speed-ups:
  - Heuristics for generating new IISs that have few overlaps with those already discovered
  - Heuristic solution of resulting integer programs
- Not used in practice

Chinneck's Heuristic: Insights

- Definitions:
  - SINF: value of elastic objective function
  - NINF: number of violated constraints
- Insights:
  - Eliminating a constraint in MIN IIS COVER should reduce SINF more than eliminating some other constraint
  - Constraints to which the elastic objective function is not sensitive do not reduce SINF when removed
  - When phase 1 ends, NINF is an upper limit on |MIN IIS COVER|. The set of violated constraints is a cover.
  - If phase 1 NINF=1, then the violated constraint constitutes a minimum cardinality IIS set cover

Chinneck's Heuristic

0. Set up elastic LP
1. Solve elastic LP
   If NINF=1, add constraint to *CoverSet* and exit.
   Candidates = {constraints to which elastic objective is sensitive}
2. For each constraint in *Candidates*:
   - Delete the constraint and solve elastic LP.
   - If NINF=0, add constraint to *CoverSet* and exit.
   - If SINF smallest, make this constraint the *winner*.
   - Reinstate the constraint.
3. Add *winner* to *CoverSet*.
   Delete *winner* permanently.
   Go to step 1.

OUTPUT: *CoverSet* is a small cardinality IIS cover.

Chinneck's Heuristic: Speed-ups

- Remember constraints that were sensitive when *winner* deleted: don't re-solve LP.
- Reduce length of candidate list:
  - *Constraint violated in elastic solution*: good predictor of the magnitude of ΔSINF due to deletion is (constraint violation) × |(constraint sensitivity)|
  - *Constraint not violated in elastic solution*: good predictor of relative magnitude of ΔSINF due to deletion is |(constraint sensitivity)|.
  - Limit candidate list to top *k* in both lists

Chinneck's Heuristic: Empirical Results

- 29 infeasible LP models from Netlib
- Original heuristic: 29/29 correct min cover
- Shorter candidate list:
  - List length 1: 25/29 correct min cover
  - List length 7: 27/29 correct min cover
  - Order of magnitude less effort

1.3 Software (1)

- MINOS(IIS) [research: from 1989]
  - *IIS isolation*: Deletion, sensitivity, elastic, reciprocal filtering and all combinations. Guide codes.
  - *MIN IIS COVER*: Chinneck's heuristics
- CLAUDIA [proprietary: from 1985]
  - Several heuristics for finding ISs
  - 1994: deletion filtering added to find IISs
- LINDO [commercial: from 1994]
  - IIS isolation via deletion filter
  - Classes IIS members as *necessary* or *sufficient*
- Cplex [commercial: from 1994]
  - Deletion/sensitivity filter for speed, elastic filter followed by deletion/sensitivity for small IISs. Row aggregation for equalities.
  - 2003: weights for guiding IIS search

Software (2)

- OSL [commercial: from 1995]
  - Deletion/sensitivity and elastic filtering
- XPress-MP [commercial: from 1997]
  - Deletion/sensitivity and elastic filtering
  - 2004: added to Mosel
- Frontline Systems [commercial: from 1997]
  - Deletion/sensitivity and elastic filtering
  - Excel add-in
- OR/MS Today LP Survey Dec. 2003
  - 27 of 44 solvers or modelling systems surveyed have infeasibility analysis capability (mostly IIS isolation)

1.4 Applications

1.4.1 Applications in Formulations

Analyzing LP Unboundedness

- primal unbounded $\Rightarrow$ dual infeasible
- IIS isolation on infeasible dual yields a "minimal unbounded set" of variables in the primal
- Available in LINDO

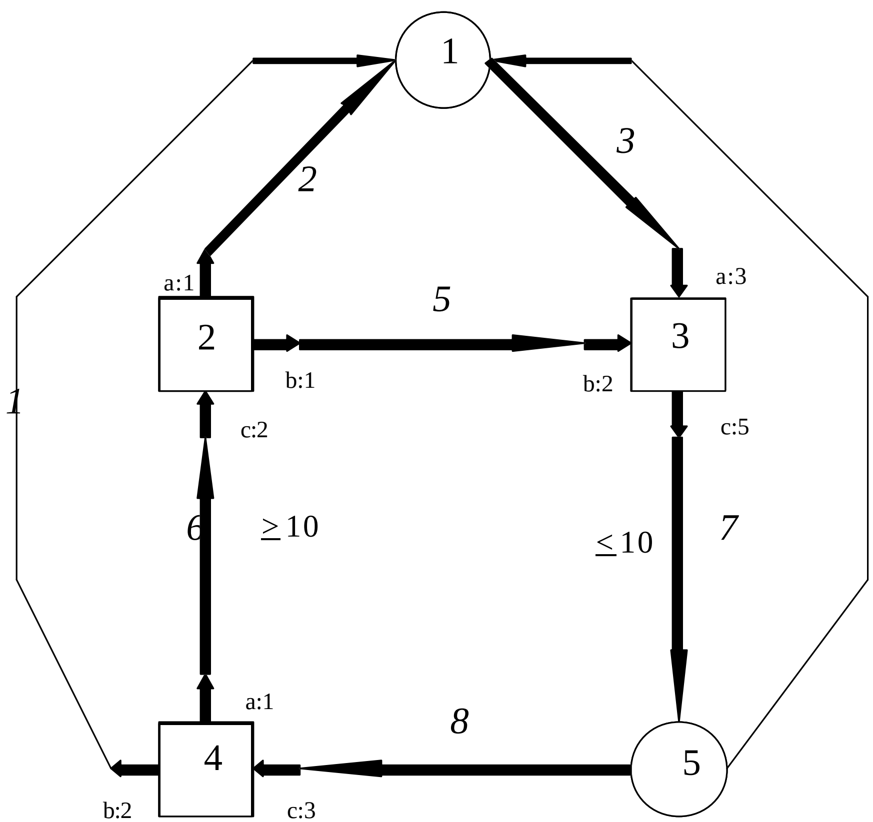
Formulating Network Models

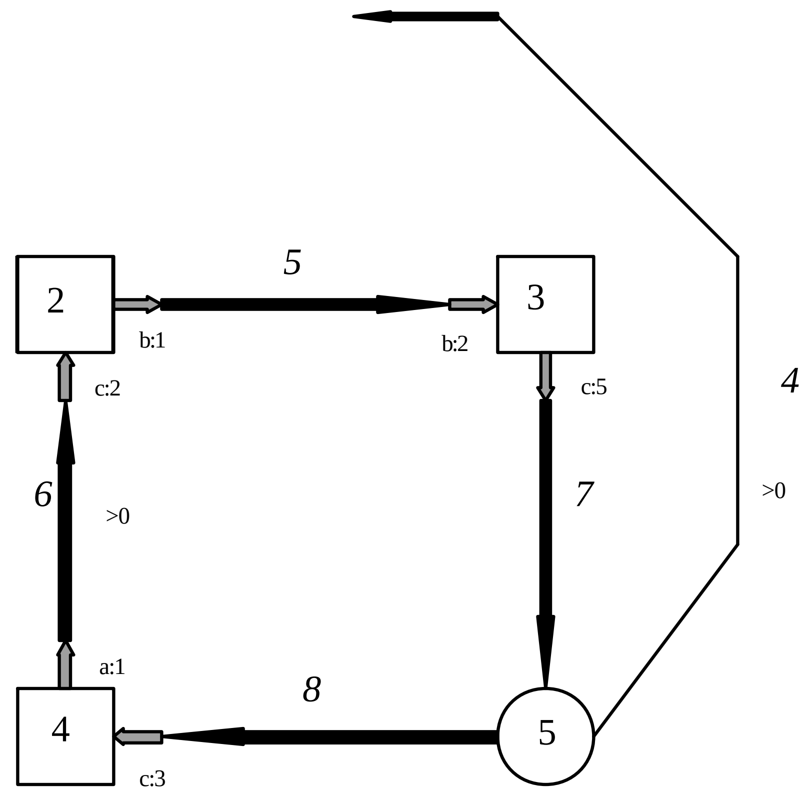
- Advanced networks:
  - generalized, processing
  - Additional structure: fixed ratios of flow at nodes
- Network *Viability*:
  - Network structure: interconnection, flow ratios, flow nonnegativity
  - *Nonviable network*: the network structure does not allow some arcs to transport any flow

IIS Isolation in Diagnosing Nonviability

- Set up viability testing LP:
  - Structural relationships (including flow ratios)
  - Creates a conic feasible region rooted at zero
  - Positivity constraint on arcs: $x_i = 1$
- If infeasible, then network is nonviable
  - IIS isolation identifies a minimal nonviability

Example





Minimal Nonviable Subset

Formulating Multiple Objective LPs

- ***True MOLP:*** at least two objectives are in conflict (optima at different extreme points).
- Types of relationships:
  - **Hard constraint:** definitely a constraint (e.g. basic physical relationship)
  - **Soft constraint:** tentatively classified as constraint, but could be an objective.
  - **Hard objective:** definitely an objective.
  - **Soft objective:** tentatively classified as objective, but could be a constraint.
- ***Aspiration level:***
  - value assigned to RHS of a soft constraint
  - RHS of soft objective when converted to constrant

MOLP Formulation Issues

- *Final Classification of soft constraints and objectives:*
  - Should a soft constraint be converted to an objective?
  - Should a soft objective be converted to a constraint, and if so, what should the aspiration value be?
- *Simplification:*
  - elimination of constraints and objectives, rewriting of constraints, resetting of aspiration values etc. to yield a simpler or clearer formulation.
  - Assigning lexicographic order to objectives

MOLP: Objectives Interaction Analysis

1. Find the extreme aspiration level for every objective:
   - Discard all objectives but one. Find its optimum value.
2. Convert each objective to a constraint:
   - Use extreme aspiration level for RHS.
3. Set up new LP that includes all constraints and all converted objectives. Solve.
4. Analyze.
   - Feasible? Not a true MOLP.
   - Find IISs. Each IIS will involve at least two conflicting objectives.

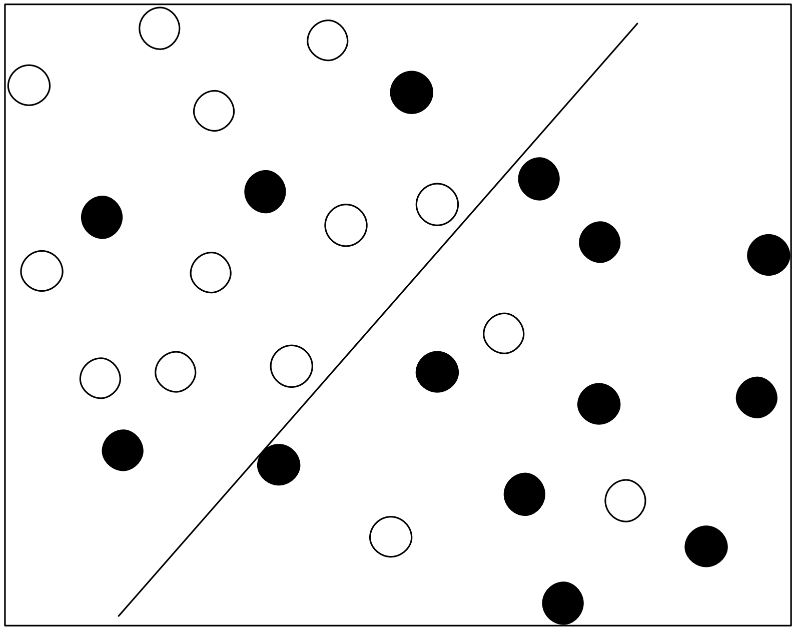
MOLP: Analysis

- IIS involves only hard constraints and converted hard objectives:
  - Abandon an objective? Set lexicographic order?
- IIS includes at least one converted soft objective or soft constraint:
  - Reformulate soft constraint or soft objective?
- Use MIN IIS COVER approach on the converted hard objectives:
  - Find fewest objectives to eliminate so that the rest can reach their aspiration levels
- Evaluate degree of interference between objectives using the *objective interference table*

1.4.2 Other Applications

Classification

- Find a hyperplane that separates two types of points with the highest accuracy
- Minimizing squared error:
  - one outlier unduly affects plane



Classification

- Find separating hyperplane $w_1x_1 + w_2x_2 + \ldots + w_Jx_J = w_0$
- Given:
  - I data points (i=1…I) in J dimensions (j=1…J)
  - d_{ij}: value of attribute j for point i
  - class of each point is known (Type 0 or Type 1).
- Define one linear inequality for each data point):
  - for each Type 0 point: $\Sigma_j d_{ij} w_j \leq w_0 - \epsilon$
  - for each Type 1 point: $\Sigma_j d_{ij} w_j \geq w_0 + \epsilon$
  - ϵ is a small positive constant (often set at 1).
  - Variables are the unrestricted w_j *and* d_{ij} are known constants.
- Solve resulting set of constraints:
  - Feasible? Points are linearly separable
  - Infeasible? MIN IIS COVER gives small(est) number of misclassified points.

Classification: Empirical Results

data set	net pts	no. features	CLIIS misclass card.	CLIIS % correct	CLIIS secs	MISMIN misclass card.	MISMIN % correct	MISMIN secs
breast cancer	683	9	**11**	98.4	17	12	98.2	0.7
pima	768	8	**149**	80.6	1662	150	80.5	1.5
bupa	345	6	**86**	75.1	159	90	73.9	0.6
wpbc	194	32	**6**	96.9	17	17	91.2	1.5
ionosphere	351	34	**6**	98.3	44	**6**	98.3	2.6
glass (type 2 vs. others)	214	9	**39**	81.8	38	50	76.6	0.6
iris (versicolor vs. others)	150	4	**25**	83.3	5	27	82.0	0.3
iris (virginica vs. others)	150	4	**1**	99.3	0.4	**1**	99.3	0.3
new thyroid (normal vs. others)	215	5	**11**	94.9	3	14	93.5	0.3

Applications in the Literature (1)

- Training neural networks:
  - Each neuron is a separating hyperplane
- Radiation Therapy Dose Planning
  - Difficult to find a feasible solution
- Design/Analysis of Protein Folding Potentials
  - IIS analysis to determine errors in approximate linear models
- Statistics:
  - Learning missing values from summary constraints

Applications in the Literature (2)

- Automatic Test Assembly
  - analysis of infeasible sets of constraints on test contents
- Backtracking in Constraint Logic Programming
  - Infeasibility encountered as constraints added
  - Backtrack in IIS order instead of ordered added
- Various NP-hard Problems:
  - Satisfiability
  - Set-covering
  - Approximability of NP-hard problems
  - Etc.

2. Faster Feasibility

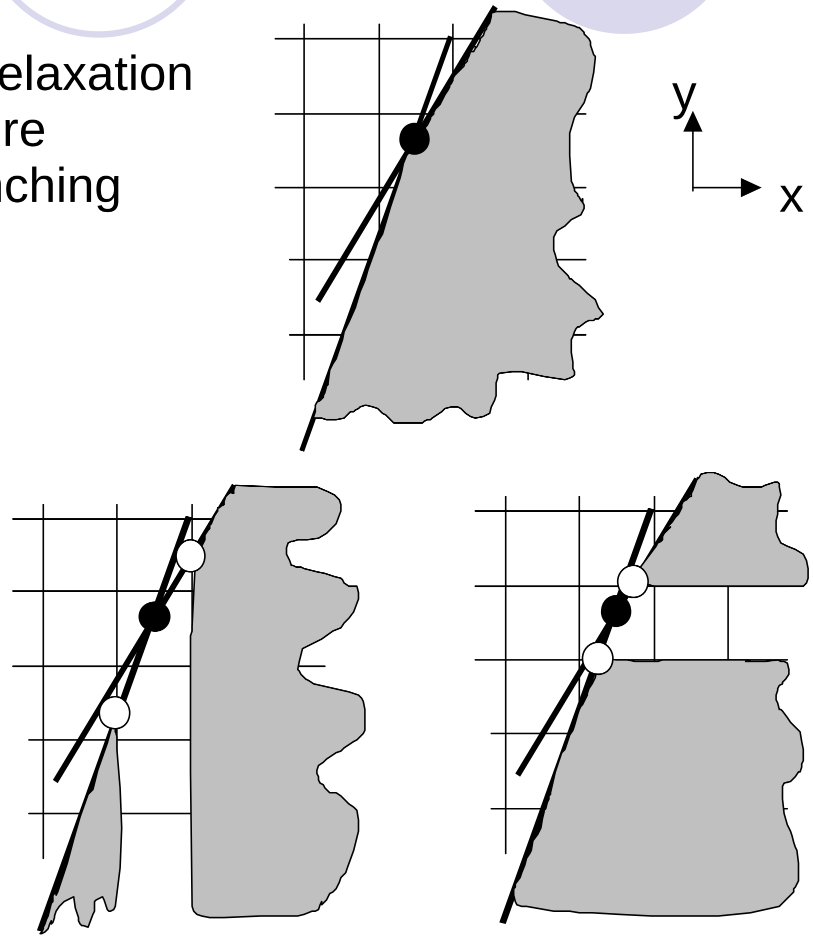
- MIP:
  - Must develop entire B&B tree to prove infeasibility.
- NLP:
  - Difficult to reach a feasible point, if one exists, reliably
- Can feasibility be reached faster?

2.1 Faster MIP Feasibility

- Branching variable selection can have a big impact on speed to first feasible solution:
  - E.g. MIPLIB *swath*: 6206 nodes (Cplex 6.5, heuristics off) vs. 27 nodes (new heuristic)
- State of the Art:
  - Select branching variable based on impact on *objective function* (pseudo-costs, etc.)
- New Idea:
  - Select branching variable based on impact on *active constraints* at parent node LP relaxation optimum

Active Constraint Variable Selection

LP relaxation before branching



Branch on x

Branch on y

Active Constraints: Overview

1. Calculate “weight” of each variable in each active constraint *(0 if the variable does not appear in constraint).*
2. For each variable, total the weights over all of the active constraints.
3. Choose variable that has the largest total weight.

- *Dynamic* variable ordering: changes at each node.

Example Weighting Schemes

If variable *i* is in active constraint *k*:

A: $W_{ik}=1$

E: W_{ik} = |coeff$_{ik}$| / [Σ|coeff of ***all*** variables|]

I: choose varb having most "votes" in A-G

O: W_{ik} = |*coeff*$_{ij}$| / (no. of int. var. in con. *k*)

- 24 new methods in all
- Tested vs. Cplex 6.5, Cplex 8.0 and OSL
  - Heuristics off, heuristics on
  - Speed metric: no. of B&B nodes
- 65 problems in MIPLIB 3.0 library

Empirical Results

	All 65 Models					50-55 Comparable Models		
method	times within 10% of best	times faster/= Cplex	FSR	times term.	QSR over non-term. models	Avg. nodes:	(avg. nodes)/ (cplex avg. nodes)	avg. ratio to best
A	**35**	42/2	0.68	5	0.42	40.18	0.1	1.81
I	29	46/2	0.74	5	0.47	37.84	**0.09**	1.61
O	33	**54/0**	**0.83**	3	**0.54**	**29.75**	0.1	**1.18**
OSL 3.0	7	21/3	0.37	2		85.62	0.21	4.63
Cplex 6.5	15			**1**		408.46		19.34
Cplex 8.0	9			**0**		310.75		10.69

2.2 Faster NLP Feasibility

Goal: given arbitrary initial point, move to a near-feasible point quickly

- Unbounded variables? Ranges too wide?

- "near-feasible"?
  - Traditional: |RHS-LHS| ≤ tolerance
    - Function scaling means this varies widely!
  - New: Euclidean distance to feasible region
    - This is a *variable-space* measure

The Constraint Consensus Method

- *Feasibility vector:* for a violated constraint, a vector indicating step to closest feasible point
  - |feasibility vector| gives distance to feasibility
  - Exact for linear constraints, approximation based on gradient for nonlinear constraints

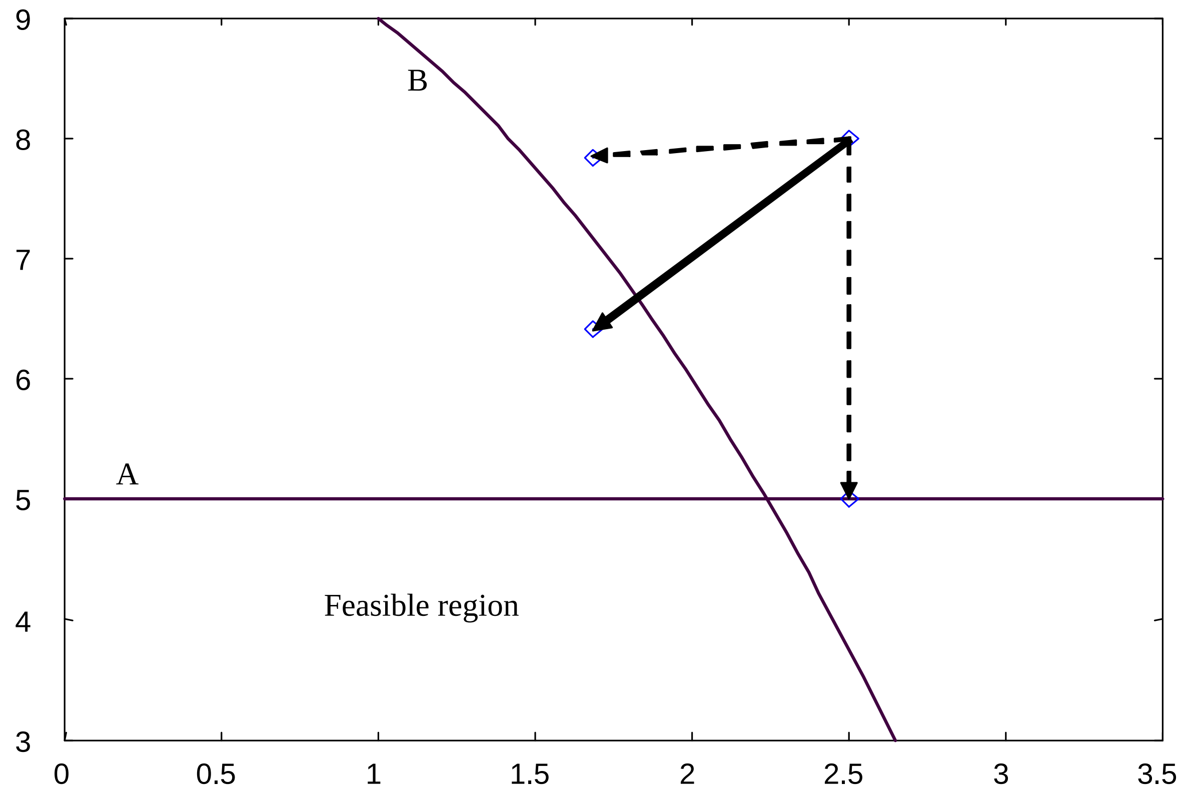
Method:

- Construct feasibility vector for each violated constraint
- Construct *consensus vector* by combining feasibility vectors in various ways
- Take the step indicated by the consensus vector
- Repeat until close enough to feasibility

Simple: no LP solutions, line search, matrix inversion, etc.

Example Constraint Consensus Step

- Next step will reach feasibility



Initial Point Heuristic

What if initial point is *not* given?

- New initial point heuristic avoids various problems:
  - If doubly bounded: set at midpoint + (small random e)
  - If single lower bound: set at bound + (small random e)
  - If single upper bound: set at bound - (small random e)
  - If unbounded both directions: set at zero + (small random e)
- Couple with CC algorithm, use to start NLP solvers
- Tested on ~230 CUTE models
  - At least one NL constraint
  - Less than 300 constraints
- Impact on NL solver ability to reach feasibility
  - MINOS, SNOPT, KNITRO, DONLP2, CONOPT

New Heuristic + CC + solver

- Using feasibility distance 0.1 for CC algorithms
- Improves over new heuristic + solver

	MINOS	SNOPT	KNITRO	DONLP2	CONOPT
modeller	*0.864*	*0.684*	*0.939*	*0.899*	*0.877*
simple	0.868	0.689	0.908	0.899	0.877
DBmax	0.864	0.693	0.912	**0.908**	0.882
DBavg	0.864	0.702	0.908	0.895	0.890
DBbnd	**0.873**	0.697	**0.921**	0.899	0.890
FDnear	0.864	0.689	0.904	0.882	0.890
FDfar	**0.873**	**0.706**	0.917	**0.908**	**0.904**

Useful Sources

General overview of state of the art:

- J.W. Chinneck (1997), “Feasibility and Viability” in *Advances in Sensitivity Analysis and Parametric Programming*, T. Gal and H.J. Greenberg (eds.), International Series in Operations Research and Management Science, Vol. 6, pp. 14-1 to 14-41, Kluwer Academic Publishers.

On constraint consensus method for NLPs:

- J.W. Chinneck (2003), “The Constraint Consensus Method for Finding Approximately Feasible Points in Nonlinear Programs”, *INFORMS Journal on Computing*, to appear.

On active constraints method for MIPs:

- J. Patel and J. Chinneck (2003), “Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs”, in review.

Other info/software:

- www.sce.carleton.ca/faculty/chinneck.html