Feasibility and Infeasibility in Optimization

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

A tutorial for CP-AI-OR-07
May 23-26, Brussels, Belgium

Why (In)feasibility is 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:
  - Backtracking in constraint logic programs
  - Training neural networks / classification
  - Radiation treatment planning
  - Statistical analysis
  - Applications to NP-hard problems
  - Protein folding …

Outline

1. Finding Irreducible Infeasible Subsets of Constraints (IISs)
   1. General Methods
   2. Linear Programming
   3. Mixed-Integer Programming
   4. Nonlinear Programming
   5. Software
   6. Constraint Programming
   7. Application to Other Model Issues
2. Finding Maximum Feasible Subsets of Constraints (Max FS)
3. Repairing Infeasibilities
4. Faster Feasibility
   1. Mixed-Integer Programs
   2. Nonlinear Programs

Analyzing Infeasible Math Programs

Three main approaches:

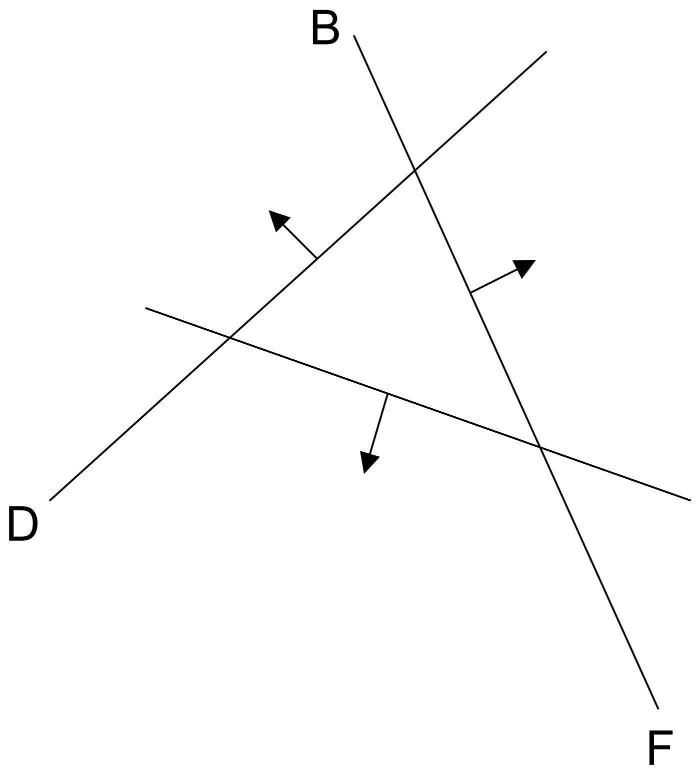
- Isolate an ***Irreducible Infeasible System (IIS)***
  - An infeasible set of constraints that becomes feasible if any constraint removed
- Find a ***Maximum Feasible Subset (Max FS)***
  - Maximum cardinality subset of constraints that is feasible
- Find **"best fix"** for infeasible constraints
  - Different matrix norms for measuring "best fix"

1. Isolating IISs

Using IISs

Cycle:

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



1.1 General Methods for Finding IISs

- Assume solver perfectly accurate in deciding feasibility status of a set of constraints
  - Reasonable assumption only for LP
- 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?
  - *For LP:* time to isolate IIS usually a small fraction of time to find infeasibility initially
    - Due to advanced starts:
      each LP is very similar to the previous one
  - *For MIP and NLP:* slow

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:
  - If IIS is small and early in the list of constraints, can use far fewer feasibility tests than deletion filter
  - *For LP:*
    speed similar to deletion filter due to basis re-use
  - *For MIP and NLP:* slow

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 (***SINF**: sum of infeasibilities*)

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 adding 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.

- {<u>A</u>,<u>B</u>,<u>C</u>,<u>D</u>,<u>E</u>,<u>F</u>,<u>G</u>} feasible. B stretched.
- {<u>A</u>,B,<u>C</u>,<u>D</u>,<u>E</u>,<u>F</u>,<u>G</u>} feasible. F stretched.
- {<u>A</u>,B,<u>C</u>,<u>D</u>,<u>E</u>,F,<u>G</u>} feasible. D stretched.
- {<u>A</u>,B,<u>C</u>,D,<u>E</u>,F,<u>G</u>} infeasible.

Output: the set {B,F,D}

  - Not necessarily an IIS until deletion filtered
  - Why? Parts of larger IISs also returned in output

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

Speed-up: Grouping Constraints

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

- Fixed group size? Adaptive group sizing?

1.2 Special Methods for LP

Bound-Tightening

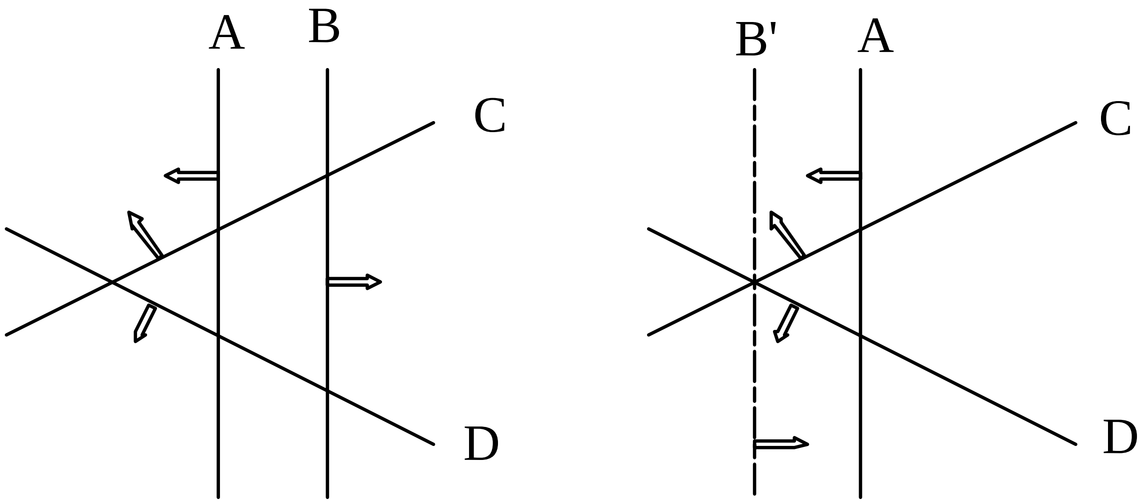
- Standard presolver techniques: iterative tightening of bounds. E.g.:
  - $2x_1 - 5x_2 \leq 10$ where $-10 \leq x_1, x_2 \leq 10$
  - Apply constraint with x_1 is at it's lower bound: $2(-10) - 5x_2 \leq 10 \Rightarrow x_2 \geq -6$.
  - Lower bound on x_2 tightened.
- May lead to detection of infeasibility.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

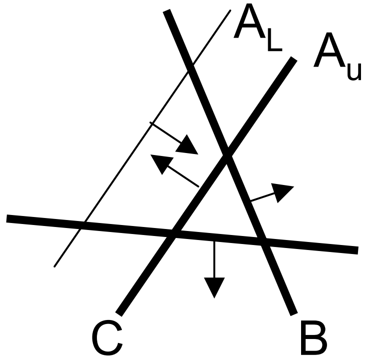
Interior Point Methods

- 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
- Characteristics:
  - Size doubles when equalities converted to inequalities
  - 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

- May have multiple IISs for *same* infeasibility
  - IISs having few row constraints preferred
- 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.
- Characteristics:
  - 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

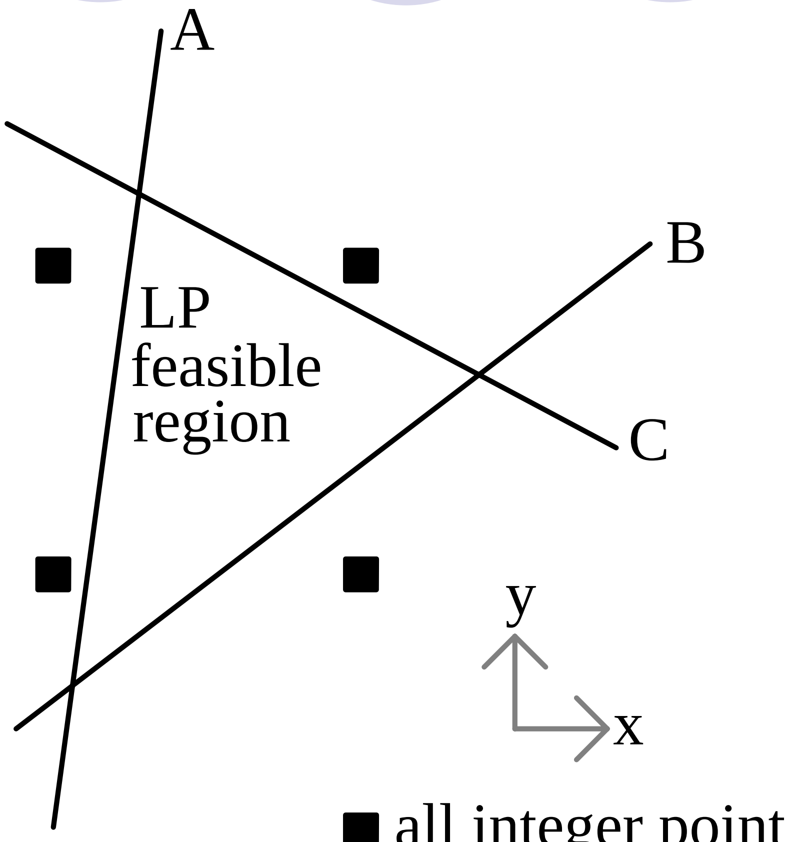
Column Bounds in the IIS:
x30 <= 12509
x50 <= 12509
x69 <= 14434
x475 <= 14434
x477 >= 0

Aggregate sum of the balance constraints:
- x30 - x50 - x69 - x475 + x477 = -60342

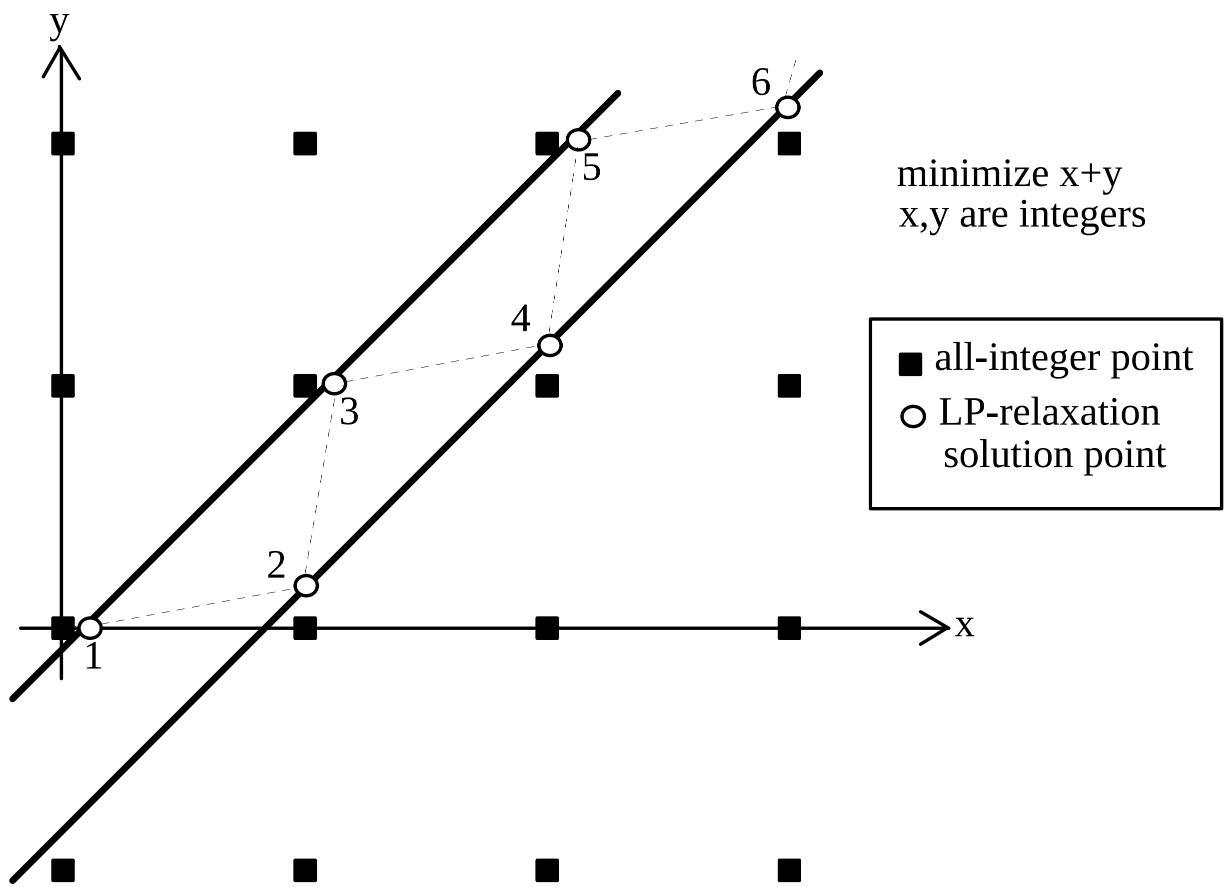
Before: 22 rows, 5 bounds, numerous variables
After: 1 row, 5 bounds, 5 variables

1.3 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
- Very slow

Using the Initial B&B Tree

Any information in original B&B tree?

- No IIS has IR set identical to the set of IRs satisfied at any intermediate node.
- Mark sensitive LCs and BDs at all leaf nodes. IR∪{marked LCs}∪{marked BDs} is infeasible.
  - Some LCs and BDs can be eliminated
- *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.

1.4 Special Methods for NLP

- NLP solvers *not* perfectly accurate in deciding feasibility.
  - *Factors:* NLP algorithm, 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 <u>report</u> infeasibility with a given set of parameter settings (including initial point, tolerances, termination conditions, etc.)
- Missing constraints can cause math errors: sqrt(x), $x \geq 0$
  - *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).

Interpretation: *this* solver finds *this* MIS intractable with *these* settings

1.5 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
  - Now discontinued
- 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)

1.6 IISs in Constraint Programming

- IIS same as:
  - Minimal conflict set
  - Minimal unsatisfiable core (MUC)
  - Minimally unsatisfiable subformula (MUS)
- IISs used for **intelligent backtracking**:
  - Backtrack only on members of IIS
  - Create new constraints based on IISs (“no-goods” learning)
- For pure logic:
  - Additive method, deletion filter, additive/deletion filter, grouping
- For mixed logic and linear constraints:
  - …add sensitivity filter, pivoting methods, advanced subsets
- For mixed logic, linear, nonlinear, integer constraints:
  - …try all above

Timeline of Early Papers (1)

- *The additive method.*
  - De Siqueira N. and Puget (1988): prototype for conjunction of clauses.
  - Tamiz, Mardle and Jones (1995, 1996): for linear programming.
  - Junker (2001): extends to general constraint programs.
- *The deletion filter.*
  - Dravnieks (1989) introduces deletion filter, sensitivity filter for LP, and elastic method. Finalized in Chinneck and Dravnieks (1991).
  - Bakker et al (1993): rediscovery for constraint satisfaction problems.
- *Pivoting methods.*
  - Gleeson and Ryan (1990)
  - De Backer and Beringer (1991): similar methods for constraint programming.

Timeline (2)

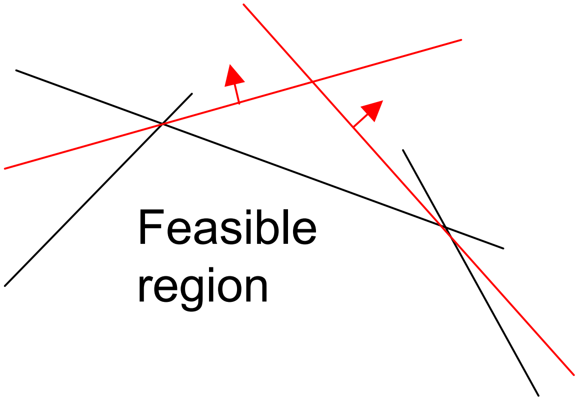
- *Constraint grouping*.
  - Chinneck (1995): initial suggestion
  - Guieu and Chinneck (1999): grouping in deletion filter and additive method for MIPs
  - Junker (2001): binary grouping for constraint satisfaction problems.
  - Atlihan and Schrage (2006) binary grouping for mathematical programs.
- *Additive/deletion filter.*
  - Guieu and Chinneck (1999): initial suggestion
  - Junker (2001): QuickXplain variants
- *Advanced subset of constraints.*
  - Guieu and Chinneck (1999): for MIP, only variables branched on form infeasible set in conjunction with their bounds and integer restrictions and the complete set of linear constraints.
  - Hemery et al (2006): the wcore concept eliminates some of the original constraints during IIS search based on the fact that they have not been used to reduce the range of any variables.

1.7 Application to Other Model Issues

- 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
- Analyzing Network Viability
  - Structural property forces all flows to zero
  - Add positivity constraint, find IIS

Other Model Issues (contd)

- Analyzing Multiple-Objective Programs
  - ***True MOLP:*** at least two objectives are in conflict (optima at different extreme points).
  - Convert objectives to constraints at their extreme points
  - Creates an infeasible LP
  - Analyze objective interactions via IIS isolation and Max FS solution



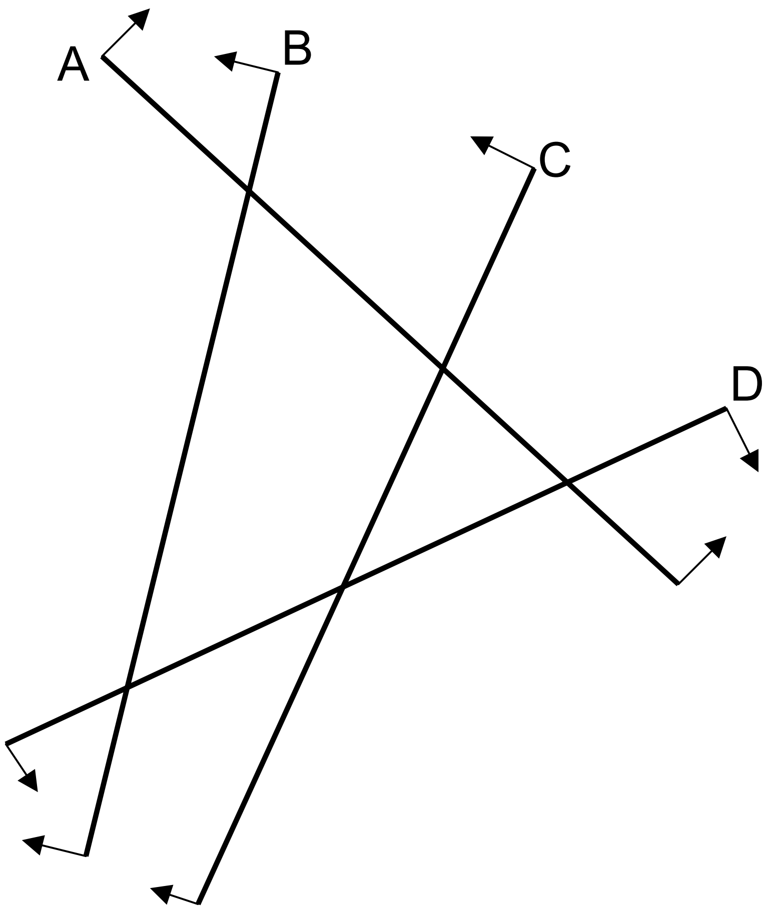
2. Finding Maximum Feasible Subsets

- Progress only on linear constraints
- Equivalent problems for an infeasible set:
  - **`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
- MIN IIS COVER is not unique

Difficulty

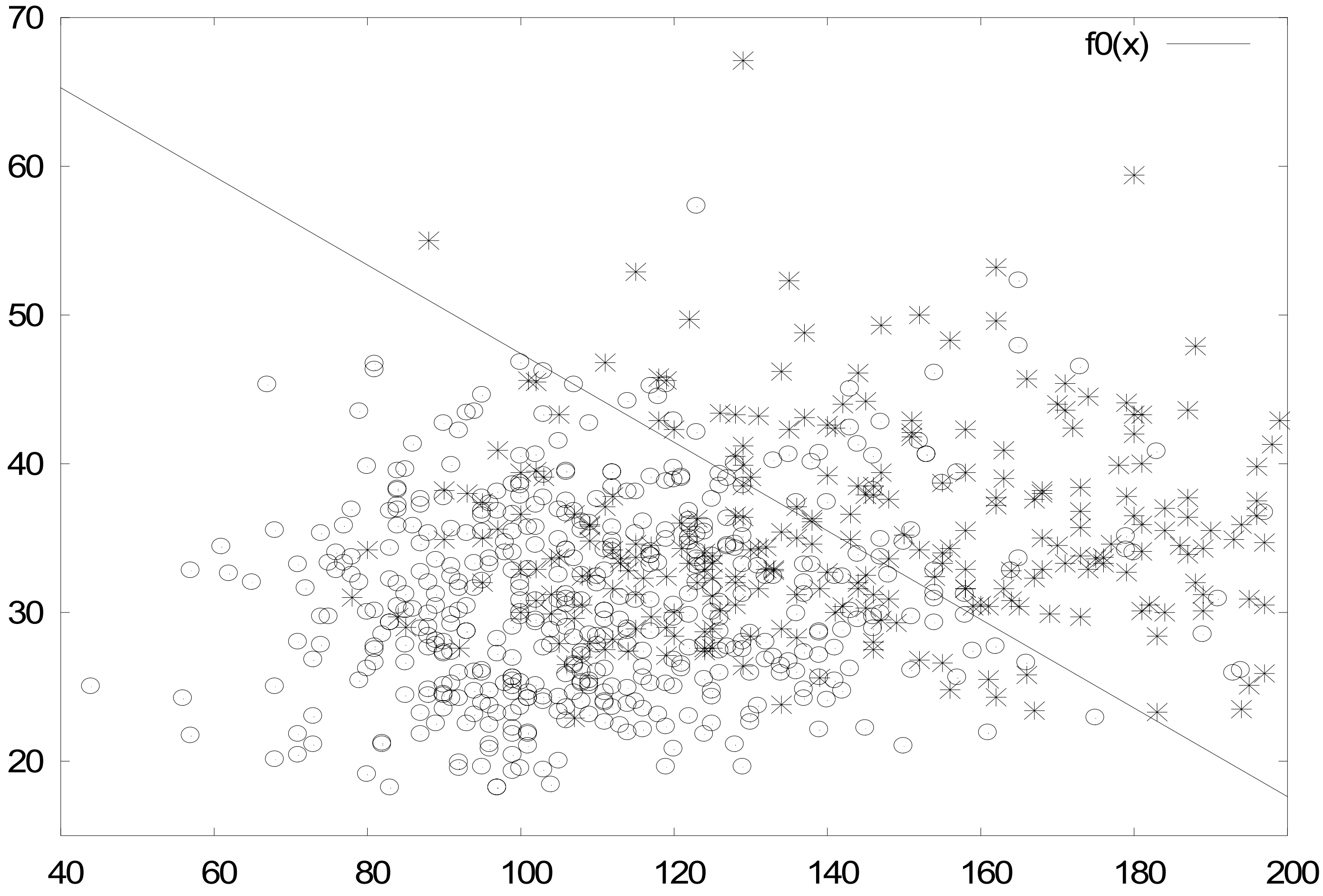
- MaxFS, MinULR, MinIISCover are NP-hard
  - Sankaran 1993, Chakravarti 1994, Amaldi and Kann 1995
- NP-hard for homogeneous systems of inequalities and binary coefficients
  - Amaldi and Kann (1995)
- Can be approximated within a factor of 2 but does not have a polynomial-time approximation scheme unless P=NP
  - Amaldi 1994, Amaldi and Kann 1995, Amaldi et al. 1999

Example



- IISs:
  {A,B,D} and {A,C,D}
- MaxFS:
  {A,B,C} or {B,C,D}
- MinULR or
  MinIISCover:
  {A} or {D}

Application: Classification, Training Neural Networks



Find separating hyperplane that misclassifies the fewest points

Classification: Convert to LP

- *Training set: I* data points in *J* dimensions
- *Attributes:* value of attribute *j* for point *i* is denoted by d_{ij}
- *Outcomes known:* either type 0 or type 1
- *Constraints:* one for each data point:
  - for points of Type 0: $\Sigma_j d_{ij} w_j \leq w_0 - \in$
  - for points of Type 1: $\Sigma_j d_{ij} w_j \geq w_0 + \in$
- *Where*
  - $\in$: small positive constant.
  - d_{ij}: known constants
  - w_j: variables, unrestricted in sign
- If LP is feasible, then data set is linearly classifiable
  - *Solution:* $\Sigma_j x_j w_j = w_0$ is classifier hyperplane

Classification: Solve via MinULR

- If ***not*** linearly classifiable (the usual case):
  - LP is infeasible
  - MinULR solution = min number unsatisfied constraints
    - *Same as min number of misclassified data points*
  - Remove constraints indicated by MinULR solution. LP now feasible
    - *Same as data set now linearly classifiable*
  - Solve feasible LP: gives eqn of separating hyperplane
    - *Same as hyperplane that misclassifies fewest points*
- Build decision trees, train neural networks

Other Applications

- Analyzing infeasible LPs
- Statistics: data depth, fixing errors, etc.
- Digital Video Broadcasting
  - Maximize regions with sufficient signal quality
- Protein Folding Potentials
  - Tens of millions of inequalities in hundreds of variables
- Radiation Therapy
  - Linear constraints and max and min radiation at locations in body
- Image and Signal Processing
- Etc.!

Exact Big-M MIP Formulation

- Minimize $Z=\Sigma y_i$ (MinULR approach)
- Subject to:
  - $\mathbf{a}_i\mathbf{x} \leq b_i + My_i$ for constraints i of type ≤
  - $\mathbf{a}_i\mathbf{x} \geq b_i - My_i$ for constraints i of type ≥
  - $\mathbf{a}_i\mathbf{x} = b_i + My_i' - My_i''$ for constraints i of type =
- Where:
  - y: binary variables
  - *M:* the usual "big-M" large positive value
- In practice:
  - Slow
  - Numerical difficulties
  - Selection of *M*

Exact LPEC Formulation

$$\max \sum_{i=1}^{m} y_i$$

$$s.t. \quad y_i \mathbf{a_i x} \{\leq, \geq, =\} y_i b_i$$

$$y_i \in \{0,1\}$$

Amaldi (2003)

- MaxFS approach
- Subst $0 \leq y_i \leq 1$without problems
- NLP solution needed

LPEC for Inequalities

$$\min \sum_{i=1}^{m} y_i$$

$$\begin{aligned} s.t. \quad & \boldsymbol{s} - \boldsymbol{Ax} + \boldsymbol{b} \geq \boldsymbol{0} \\ & \boldsymbol{y}(\boldsymbol{s} - \boldsymbol{Ax} + \boldsymbol{b}) = \boldsymbol{0} \\ & -\boldsymbol{y} + \boldsymbol{1} \geq \boldsymbol{0} \\ & \boldsymbol{s}(-\boldsymbol{y} + \boldsymbol{1}) = \boldsymbol{0} \\ & \boldsymbol{x} \in \Re^n, \boldsymbol{s} \geq 0, \boldsymbol{y} \geq 0 \end{aligned}$$

- Machine learning formulation
  - Mangasarian (1994)
- For inequalities $\boldsymbol{Ax} \leq \boldsymbol{b}$
- y_i=0 only when s_i=0
- NLP solved by successive LP

Approximate Parametric NLP

$$\min \sum_{i=1}^{m} (1 - e^{-\alpha z_i})$$

$$s.t. \quad \boldsymbol{z} \geq \boldsymbol{Ax} - \boldsymbol{b}$$

$$\boldsymbol{z} \geq \boldsymbol{0}$$

- When z_i=0, then $\boldsymbol{a}_i\boldsymbol{x} \leq b_i$ is satisfied
- α: control parameter
- NLP solved by successive LP
  - Mangasarian (1996)
  - Bennett and Bredensteiner (1997)

IIS Enumeration and Covering

- Parker (1995) Parker and Ryan (1996):
  - Enumerate IISs one at a time
    - Efficient algorithm for this
  - Solve MinIISCover each time a new IIS is found
    - By exact MIP or by greedy heuristic
    - Pfetsch (2002, 2005), Codato and Fischetti (2004, 2006): special cuts to solve MIP faster
- Tamiz (1995)
  - Heuristic enumeration of IISs
  - Frequency based heuristic to solve MinIISCover

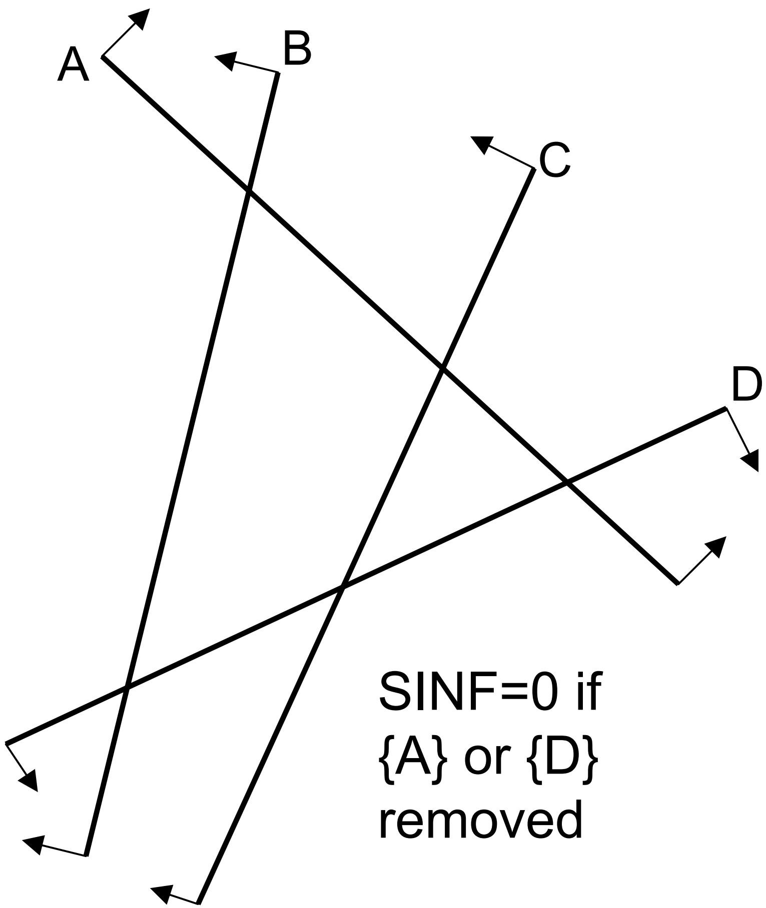
Phase 1 Heuristics

- Solve phase 1 LP:
  - ***Observation***: violated constraints constitute an IIS set cover (Chinneck 1996)
  - ***Observation:*** if just 1 violated constraint in phase1 then ***minimum*** cardinality IIS set cover (Chinneck 1996)
- Phase 1 and elastic programs:
  - *Simple phase 1*: ≥ and = constraints elasticized
  - *MINOS phase 1*: min *number* of violations and Σs_i
  - *Standard elastic program*: bounds not elasticized
  - *Full elastic program*: bounds also elasticized

Phase 1 IIS Covers

Model	Minimum cover cardinality (Parker and Ryan 1996)	MINOS Phase 1 cover	Standard elastic pro-gram Phase 1 cover
bgdbg1	12	23	13
bgindy	1	14	**1**
bgprtr	1	2	2
chemcom	1	11	12
cplex1	1	211	212
gran	not calculated	244	473
greenbea	2	3	**2**
itest2	2	**2**	**2**
itest6	2	3	4
klein2	1	3	5
klein3	1	4	19
mondou2	3	**3**	5
reactor	1	3	2
refinery	1	3	6
woodinfe	2	**2**	**2**

Chinneck's Elastic Heuristics (1996)



- ***Observation:*** Eliminating constraint in minimum-cardinality IIS set cover should reduce SINF *<u>more</u>* than eliminating constraint that is not in minimum-cardinality IIS set cover

Chinneck's Elastic Heuristics

- ***Observation:*** Constraints to which the elastic objective function is *not* sensitive do *not* reduce SINF when removed from the model
- ***Algorithm:***

1. For every constraint having nonzero dual price:
   - Temporarily remove and note new SINF
2. Permanently remove constraint that gives lowest new SINF
3. If feasible, then stop. Else go to Step 1.

Chinneck's Elastic Heuristics

- Details:
  - Full elastic model recommended
  - Remember nonzero dual prices for best new SINF seen so far: do not have to recalculate
  - If *NumInfeasibilities* = 1 during test, remember this and use it directly in next round
  - Many LPs solved
    - LP hot starts make soln times reasonable

Chinneck's Elastic Heuristics (2001)

- ***Observation:*** for constraints that are violated during phase 1 a good predictor of SINF drop when removed is (constraint violation) $\times$ |dual price|.
- Small study:
  - over 95% accurate in 87% of cases
  - over 90% accurate in 94% of cases

Chinneck's Elastic Heuristics

- ***Observation:*** for constraints that are satisfied during phase 1 a good predictor of the *relative magnitude* of the SINF drop when removed is |dual price|.

- Use these two observations to shorten length of candidate list

Chinneck's Elastic Heuristics

- ***Revised algorithm:***

1. Sort violated constraints by violn×|dual price|,
   sort satisfied constraints by |dual price|
2. For top *k* constraints in violated and satisfied lists:
   - Temporarily remove, re-solve LP, note new SINF
3. Permanently remove constraint giving lowest new SINF
4. If feasible, then stop. Else go to Step 1.

Chinneck's Elastic Heuristics

- Details:
  - Alg 2(k): just top k violated constraints
  - Alg 3(k): top k violated, top k satisfied
    - Solve at most $2k$ LPs at each iteration
  - Alg 4(k):
    - May use ordinary phase 1 to detect initial infeasibility, then full elastic phase 1 thereafter
    - Keep IIS covers from both phase 1 solns as backups
    - Otherwise same as Alg 3(k)
- Trade-off: speed vs. accuracy
- *Testing:* Alg 3(7) and 4(7) almost as good as original and an order of magnitude faster

Classification data sets

Data set	Original Alg. 1		Algorithm 2(1)		MISMIN	
	% acc.	*secs*	% acc.	*secs*	% acc.	*secs*
breast cancer	**98.4**	*17*	**98.4**	*4.3*	98.2	***0.7***
bupa	75.1	*159*	**75.9**	*1.3*	73.9	***0.6***
glass (type 2 vs. others)	**81.8**	*38*	78.5	***0.6***	76.6	***0.6***
ionosphere	**98.3**	*44*	**98.3**	*5.4*	**98.3**	***2.6***
iris (versicolor vs. others)	**83.3**	*5*	**83.3**	***0.2***	82.0	*0.3*
iris (virginica vs. others)	**99.3**	*0.4*	**99.3**	***0.1***	**99.3**	*0.3*
new thyroid (normal vs. others)	**94.9**	*3*	**94.9**	***0.3***	93.5	***0.3***
pima	**80.6**	*1434*	80.2	*7.2*	80.5	***1.5***
wpbc	**96.9**	*17*	**96.9**	***1.2***	91.2	*1.5*
average:	**89.8**	*216.2*	89.5	*2.3*	88.2	*0.9*

- MISMIN: parametric NLP approximation
  - (Bennett and Bredensteiner 1997)

Two-Phase Relaxation Heuristic

- **First phase:** heuristic soln for MaxFS
  - A linearization of the big-*M* formulation, or
  - A linearization of the bilinear LPEC, or
  - LP phase 1 heuristic
- **Second phase:** add constraints to first phase soln using more exact method:
  - Exact “big-M” MIP on smaller 2nd phase
- Amaldi, Bruglieri, Casale 2007

Two-Phase Relaxation Heuristic

$$\max \sum_{i=1}^{m} y_i$$

$$subject\ to:$$

$$\sum_{j:a_{ij}<0} a_{ij} z_{ij} + \sum_{j:a_{ij}\geq 0} a_{ij} x_j \geq y_i b_i,\ i=1\ldots m$$

$$z_{ij} \leq u_j y_i,\ i=1\ldots m,\ j=1\ldots n,\ s.t.\ a_{ij}<0$$

$$z_{ij} \leq x_j,\ i=1\ldots m,\ j=1\ldots n,\ s.t. a_{ij}<0$$

$$x_j - u_j(1-y_i) \leq z_{ij},\ i=1\ldots m,\ j=1\ldots n,\ s.t.\ a_{ij}<0$$

$$l_j \leq x_j \leq u_j,\ j=1\ldots n$$

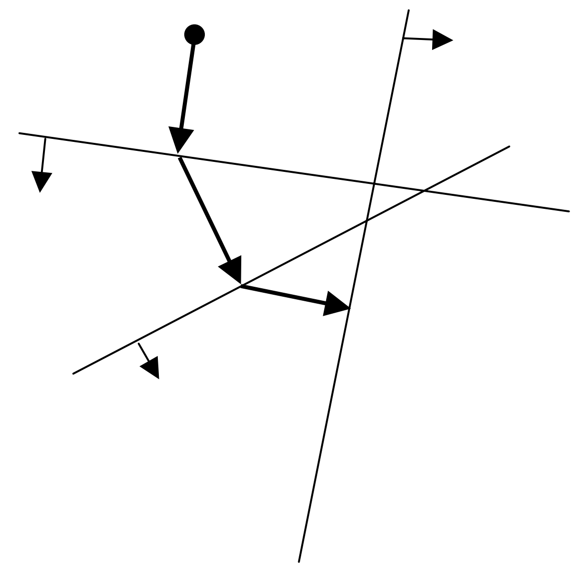
$$0 \leq y_i \leq 1, i=1\ldots m$$

$$z_{ij} \geq 0, i=1\ldots m,\ j=1\ldots n,\ s.t.\ a_{ij}<0$$

- Linearization of LPEC
- bounded inequalities only
- u_i, l_i: bounds
- $z_{ij} = y_i x_j$
- Generally best results

Randomized Thermal Relaxation

- For *very* large systems.
  - Tens of millions of inequalities
  - Amaldi, Belotti, Hauser 2005
- Thermal perceptron heuristic (Frean 1992), or sequential projection algorithm (Censor et al 2001)
  - $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \eta_i a_{ki}$ with probability p_i if constraint k_i is violated, or $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i$ otherwise
  - Select constraints with large violations near start; select constraints with small violations near end. Control by temperature parameter as in simulated annealing.
  - *Variations:* how adjust η_i, p_i, constraint selection, etc.



Randomized Thermal Relaxation

- Digital Video Broadcasting data sets:
  - Solves many more problems that Big-M MIP
- Protein-folding potentials tests:
  - Up to 837,802 rows, 301 cols
  - Cplex 8.1 Big-M unable to solve any
  - 6 very large feasible instances
    - RTR comes within 6 constraints of total number in all cases

3. Repairing Infeasibility

How to define the "best fix"?

- Shifting constraints: smallest number
  - same as Max FS, weighted or unweighted
- Shifting constraints: smallest total penalty
  - same as minimizing l_1 norm
  - same as elastic program, weighted or unweighted
- Altering constraint body: minimize a matrix norm
  - l_1 or l_∞, Frobenius norms

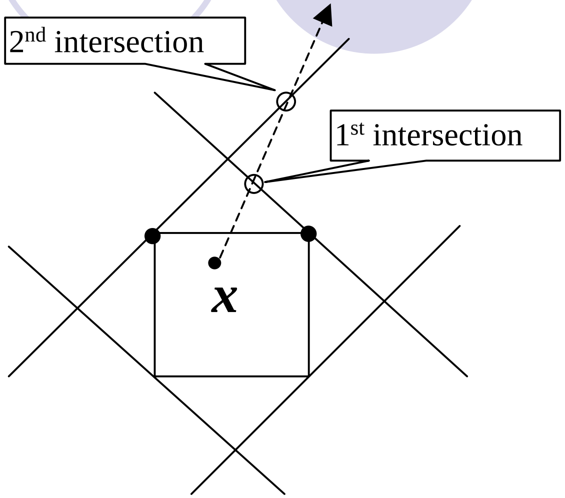
Solution methods:

- Phase 1 solution
- Elastic programs
- Max FS algorithms
- Least-squares
- Fuzzy methods
- Goal programming
- Reformulation-Linearization-Convexification
- Etc….

4.1 Faster MIP Feasibility

Methods to date:

- Pivot-and-complement for BIP
  - BIP: max $\boldsymbol{cx}$ s.t. $\boldsymbol{Ax \le b}$, x_j binary
  - LP: max $\boldsymbol{cx}$ s.t. $\boldsymbol{Ax+y=b}$, $\boldsymbol{0 \le x \le 1}$, $\boldsymbol{y \ge 0}$, y_i basic
  - Pivot to make y_i basic. Flip varbs when stuck
- Pivot-and-shift variant for MIP
- OCTANE for BIP
  - Ray in improving direction passes through extended facets of hyper-octagon surrounding hypercube of solns
  - Hit facets of octagon indicate binary solns to try in order
- The feasibility pump
  - After solving intial LP-relaxation, alternate between (i) rounding to get integer-feasible soln (that violates constraints) and (ii) nearest LP-feasible soln (that violates integrality).
  - Randomization if get stuck.



Branch and bound

Steps after start-up:

1. If no unexplored nodes left then exit: optimal or infeasible.
2. Choose unexplored node for expansion and solve its LP relaxation.
    - Infeasible: discard the node, go to Step 1.
    - Feasible and integer-feasible: check for new incumbent, go to Step 1.
3. ***Choose branching variable*** in current node and create two new child nodes.

Is Branching Variable Selection Important?

	B&B nodes to First Feasible Soln	
model	**Cplex 9.0**	**Active-Constraints Method**
aflow30a	23481	22 (A, H_M, H_O, O, P)
aflow40b	100,000+ (limit)	33 (H_O, O, P)
fast0507	14753	26 (A)
glass4	7940	62 (A, H_M, H_O, O, P)
nsrand-ipx	3301	18 (H_M)
timtab2	14059	100,000+ (limit)

Traditional Branching Variable Selection

- Based on estimated impact on ***objective function***
- *Goal:* maximize degradation in the objective function value at optimal solution of child node LP relaxations.
- e.g. pseudo-costs

New: Active Constraints Approach

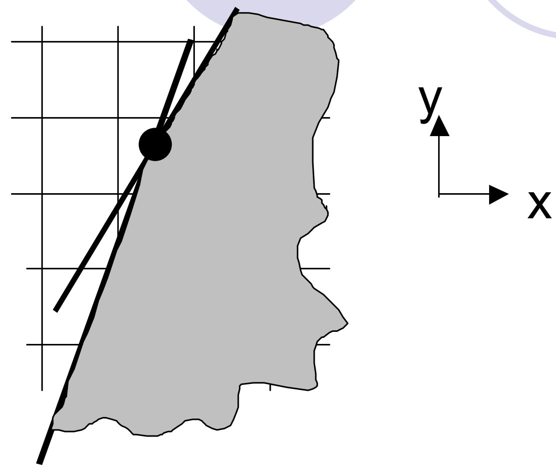
Goal: make child node LP-relaxation optima far from parent node LP-relaxation optimum.

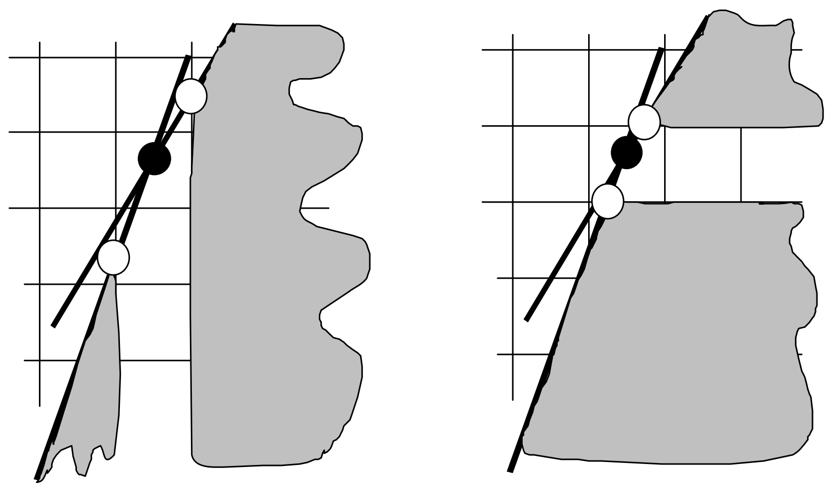
- *Active constraints* fix the position of the LP optimum solution in parent, so…
- Choose branching variable that has *most impact on the active constraints* in parent LP relaxation optimum solution.
- ***Constraint-oriented*** approach [Patel and Chinneck 2006]

Note: "active constraints" include tight degenerate constraints

Impact of the Branching Variable

LP relaxation before branching





Branch on x

Branch on y

Estimating Candidate Variable Impact on Active Constraints

1. Calculate the "weight" W_{ik} of each candidate variable i in each active constraint k
   - *0 if the variable does not appear in constraint*
2. For each variable, total weights over all active constraints.
3. Choose variable that has the largest total weight.

Dynamic variable ordering: changes at each node.

Overview of Weighting Methods

- Is candidate variable in active constraint or not?
- Relative importance of active constraint:
  - Smaller weight if more candidate or integer variables: changes in other variables compensate for changes in selected variable.
  - Normalize by absolute sum of coefficients.
- Relative importance of candidate variable within active constraint:
  - Greater weight if coefficient size is larger: candidate variable has more impact.
- Sum weights over all active constraints? Look at biggest impact on single constraint?
- Etc.

Methods A, B, L

Numerous variants. Subset of best:

- **A:** W_{ik}=1.
  - *Is candidate variable present in the active constraint?*
- **B:** W_{ik} = 1/ [Σ(|coeff of *all* variables|].
  - *Like A, but relative impact of a constraint normalized by absolute sum of coefficients*
- **L:** W_{ik} = 1/(no. *integer variables*)
  - *Like A, but relative impact of a constraint normalized by number of integer variables it contains*
  - *Related to MOMS rule?*

Methods M, O, P

- **M:** W_{ik} = 1/(no. candidate variables)
  - *Like A, but relative impact of a constraint normalized by number of candidate variables it contains*
  - *Not used directly: see H methods*
- **O:** $W_{ik} = |\text{coeff}_i|$/(no. of *integer variables*)
  - *Like L, but size of coefficient affects weight of varb in constraint*
- **P:** $W_{ik} = |\text{coeff}_i|$/(no. of *candidate variables*)
  - *Like M, but size of coefficient affects weight of varb in constraint*

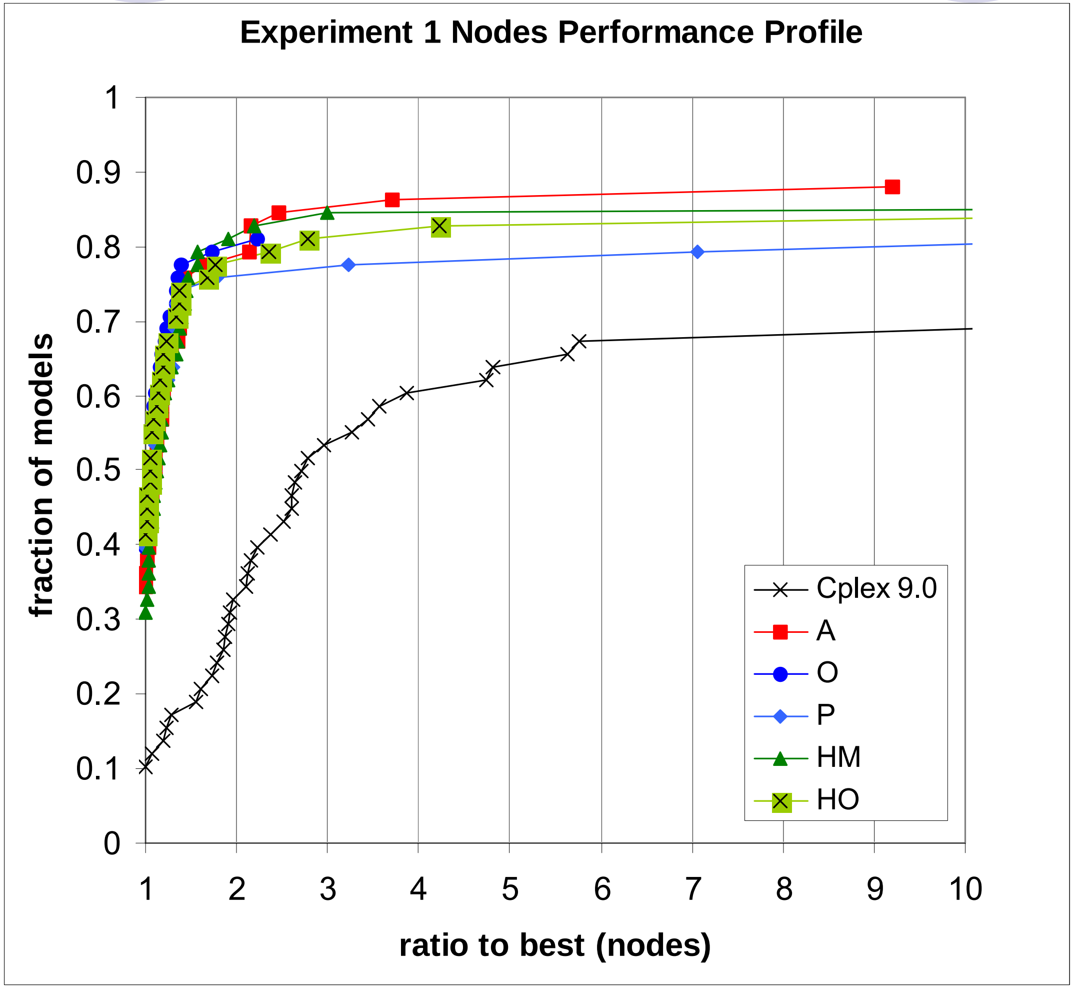
Methods H_M, H_O

- **H** methods: for a given base method, choose the variable that has largest weight in any *single* active constraint
  - Do not sum across active constraints
- $\mathbf{H_M}$: based on method M
- $\mathbf{H_O}$: based on method O

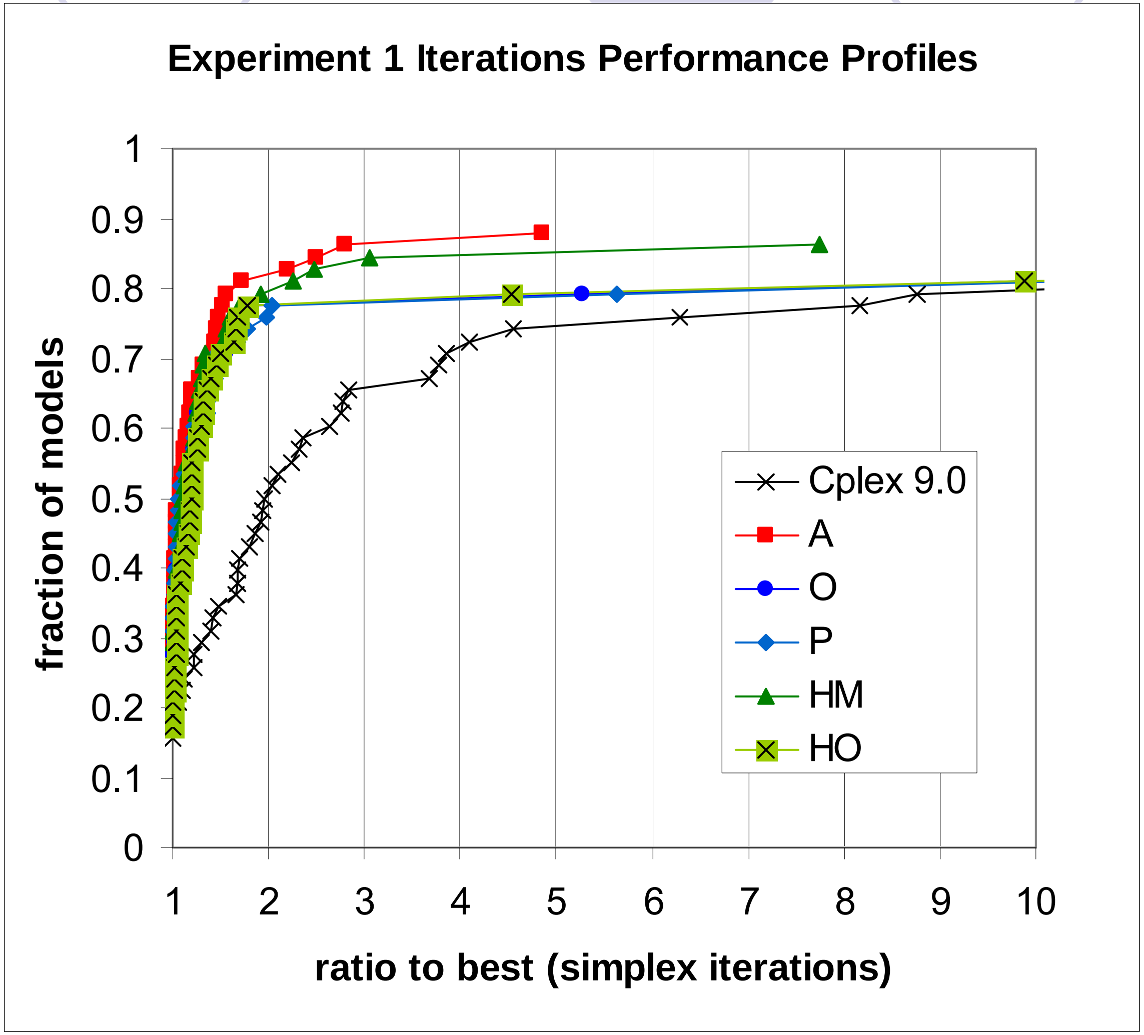
Experiments

- Solver built on Cplex 9.1
- Limits on time, nodes, node file size
- MIPLIB 2003 models
- *Experiment 1 (basic B&B)*: all heuristics **off**
- *Experiment 2*: all heuristics turned **on**

Exp 1: Nodes Peformance Profiles



Exp 1: Simplex Iterations Perf. Profiles



Experiment 2: Notes

- All internal heuristics *on*.
- Heuristics impact is mixed:
  - Many models solved at root node
  - Others: using Cplex alone:
    - half slower with heuristics on, half faster.
    - 1 model solvable with heuristics off, but not solvable with heuristics on
- Active constraints methods generally better than Cplex

Quality Success Ratios

Experiment 1 over 40 comparable models	
method	QSR
A	0.53
H_M	0.55
H_O	0.58
O	0.70
P	**0.78**

Experiment 2 over 12 comparable models	
method	QSR
B	**0.75**
H_M	0.50
H_O	0.50
L	0.58
P	0.33

4.2 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
- *Consensus vector:* update step from combination of feasibility vectors

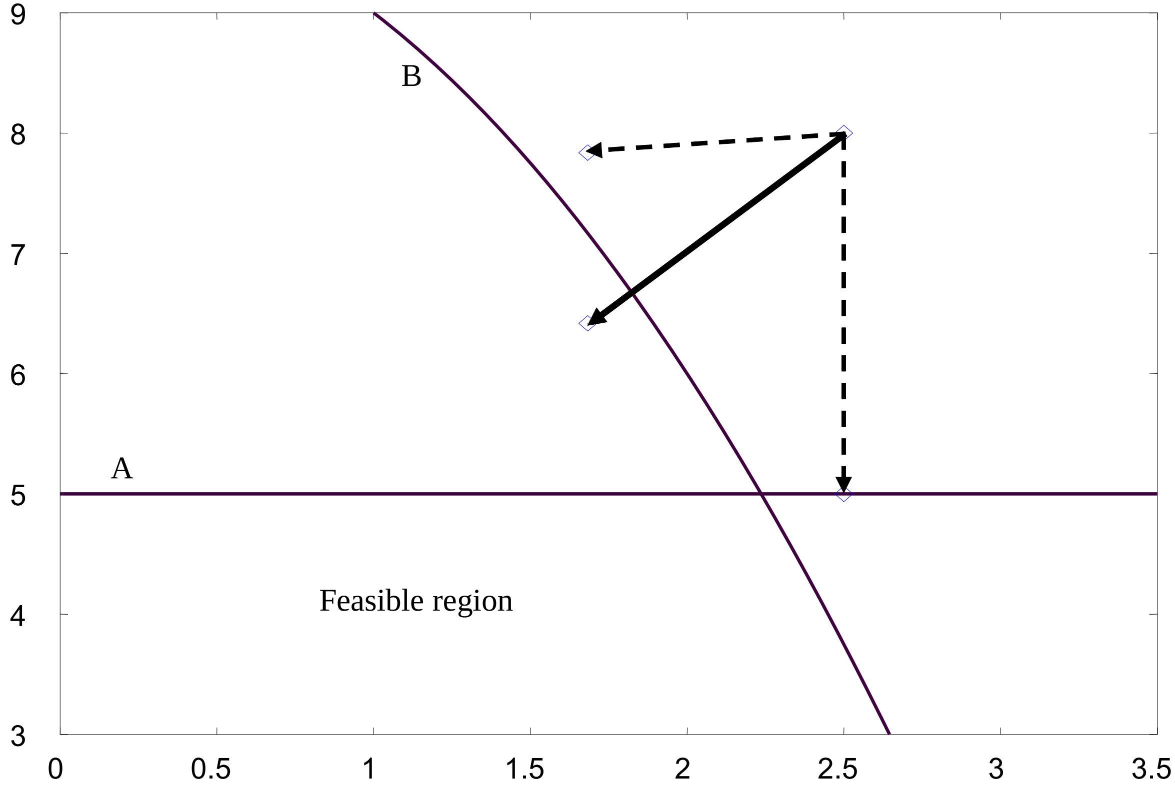
Method:

- Construct feasibility vector for each violated constraint
- Construct *consensus vector* (various options)
- 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



Where to start? 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 ε)
  - If single lower bound: set at bound + (small random ε)
  - If single upper bound: set at bound - (small random ε)
  - If unbounded both directions: set at zero + (small random ε)
- 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 (2007), “Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods”, Springer, in preparation.
- 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:

- W. Ibrahim and J.W. Chinneck (2005), "Improving Solver Success in Reaching Feasibility for Sets of Nonlinear Constraints", Computers and Operations Research, to appear

On active constraints method for MIPs:

- J. Patel and J. Chinneck (2006), “Active-Constraint Variable Ordering for Faster Feasibility of Mixed Integer Linear Programs”, Mathematical Programming, to appear.

Other info/software:

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