Decision Procedures
An Algorithmic Point of View

Deciding ILPs with Branch & Bound

ILP References:

'Integer Programming' / Laurence Wolsey

'Intro. To mathematical programming' / Hillier, Lieberman

We will see…

- Solving a linear (continues) system
  - Good old Gaussian Elimination for linear equations.
  - Feasibility test a-la Simplex for linear inequalities.
  - Fourir-Motzkin for linear inequalities.
- Solving a linear (discrete) system
  - Branch and Bound for integer linear inequalities.
  - The Omega-Test method for integer linear inequalities.

Integer Linear Programming

- Problem formulation

  max cx

  $Ax \leq b$

  $x \geq 0$ and integer

  Where A is an $m \times n$ coefficients matrix

  c is an n-dimensional row vector

  b an m - dimensional column vector

  x an n - dimensional column vector of variables.

Feasibility of a linear system

- The decision problem associated with ILP is NP-hard.
- But once again… we are not actually interested in ILP: we do not have an objective…
- All we want to know is whether a given system is feasible.

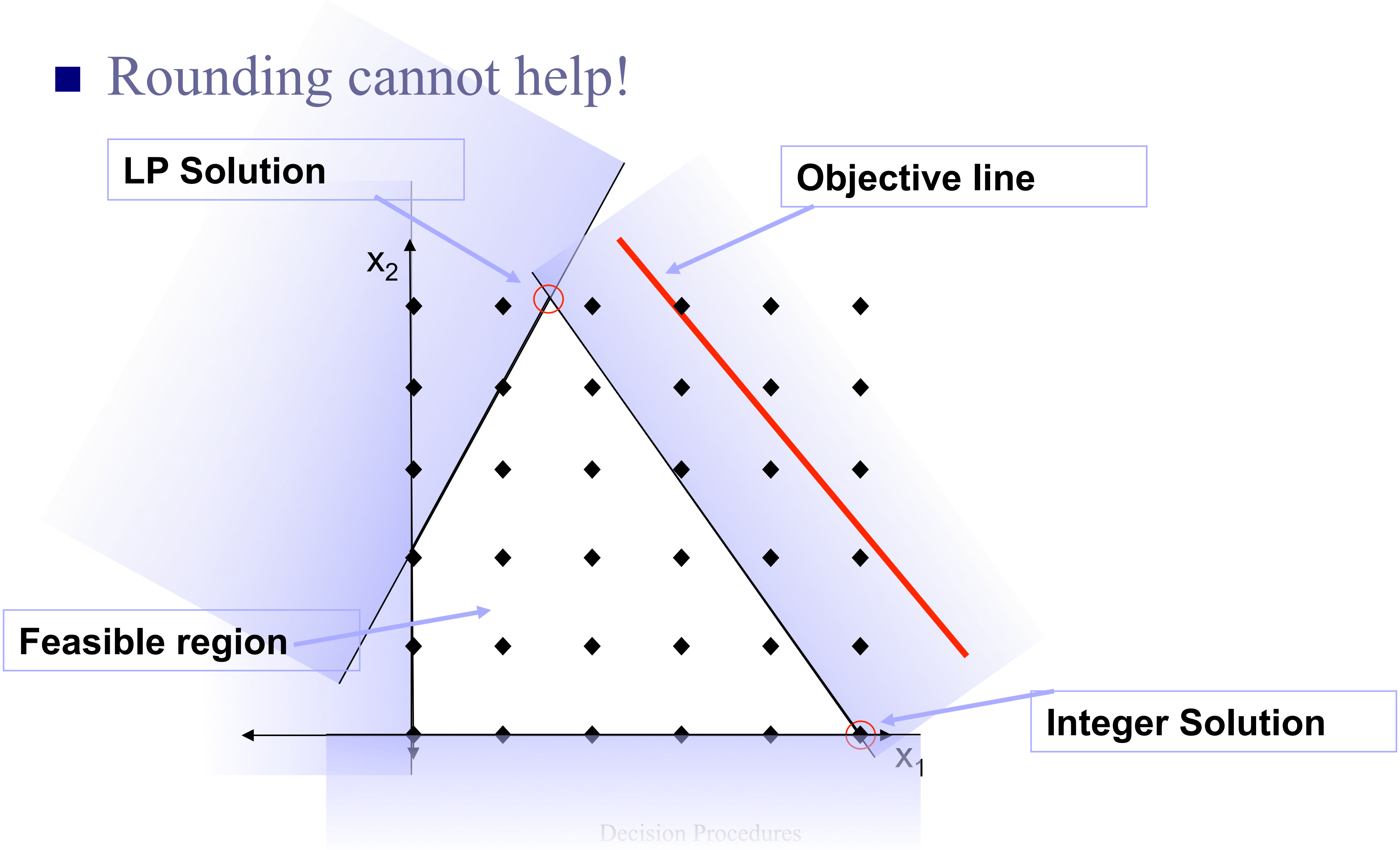
$$Ax \leq b$$

$x \geq 0$ and integer

- Still, NP-hard…

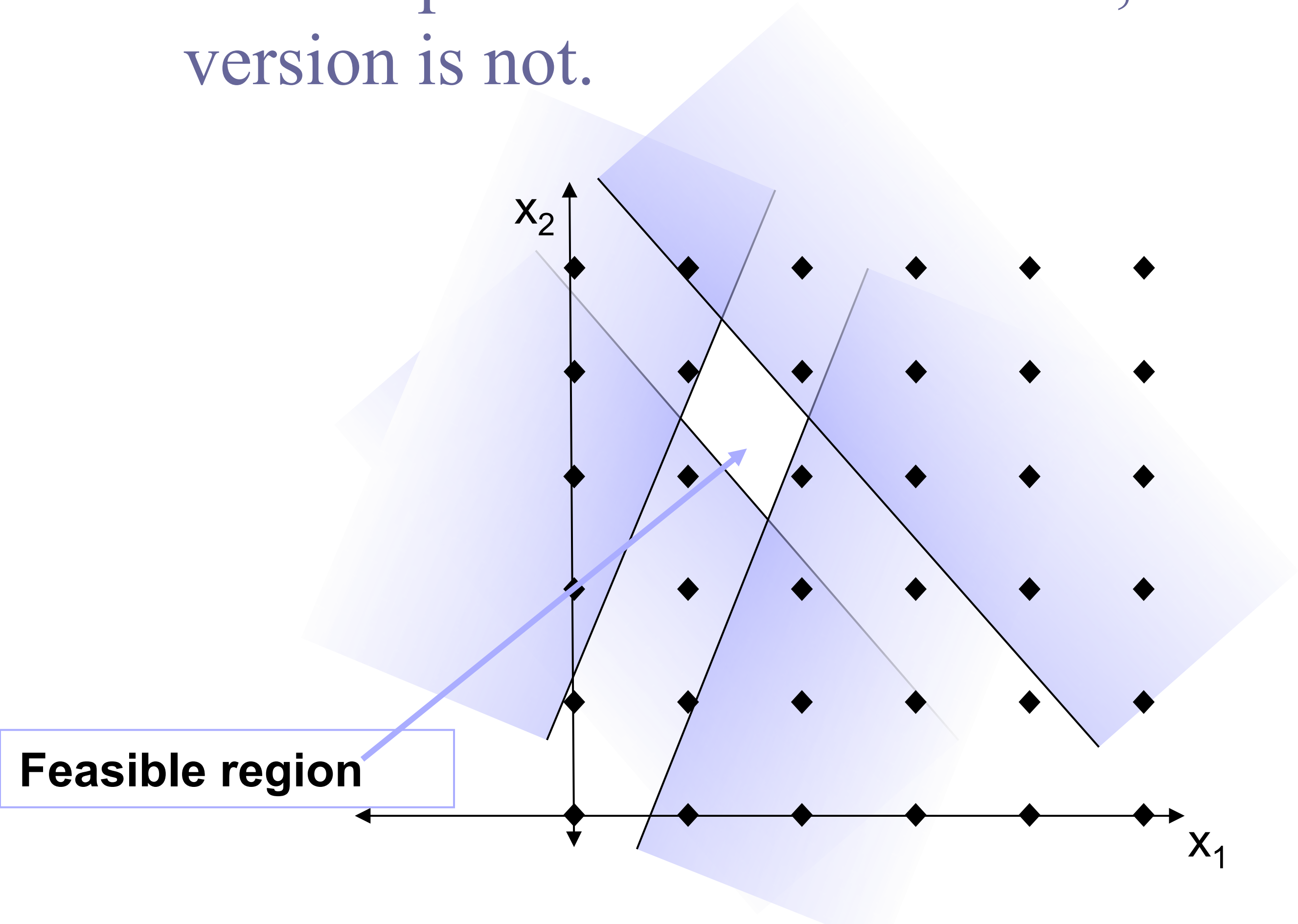
How different can it be from LP ?

- Rounding cannot help!



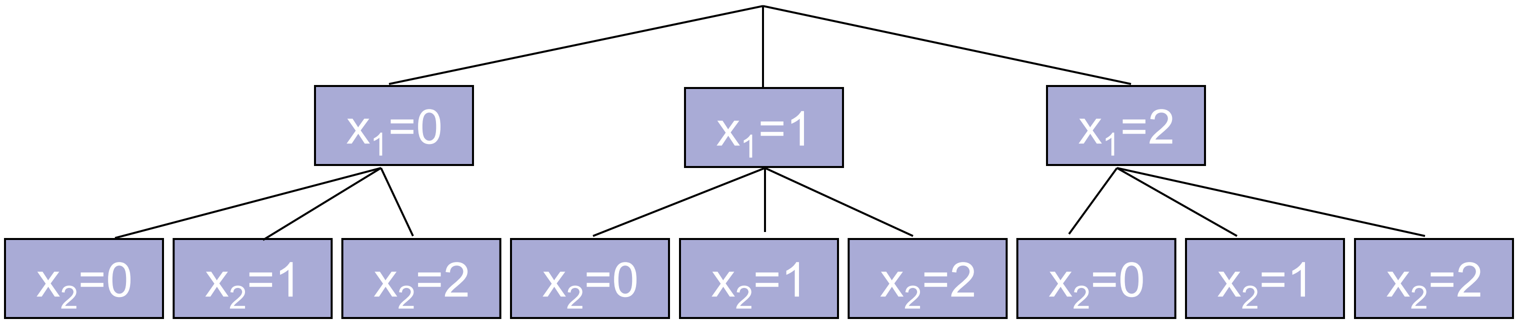
How different can it be from LP ?

- The LP problem can be feasible, whereas its ILP version is not.



A naïve solution strategy

- From hereon we will assume that all variables are finite.
- Enumerate all solutions with a tree



- Guaranteed to find a feasible solution if it exists
- But, exponential growth in the size of the tree / computation time

A family of algorithms: Branch & Bound

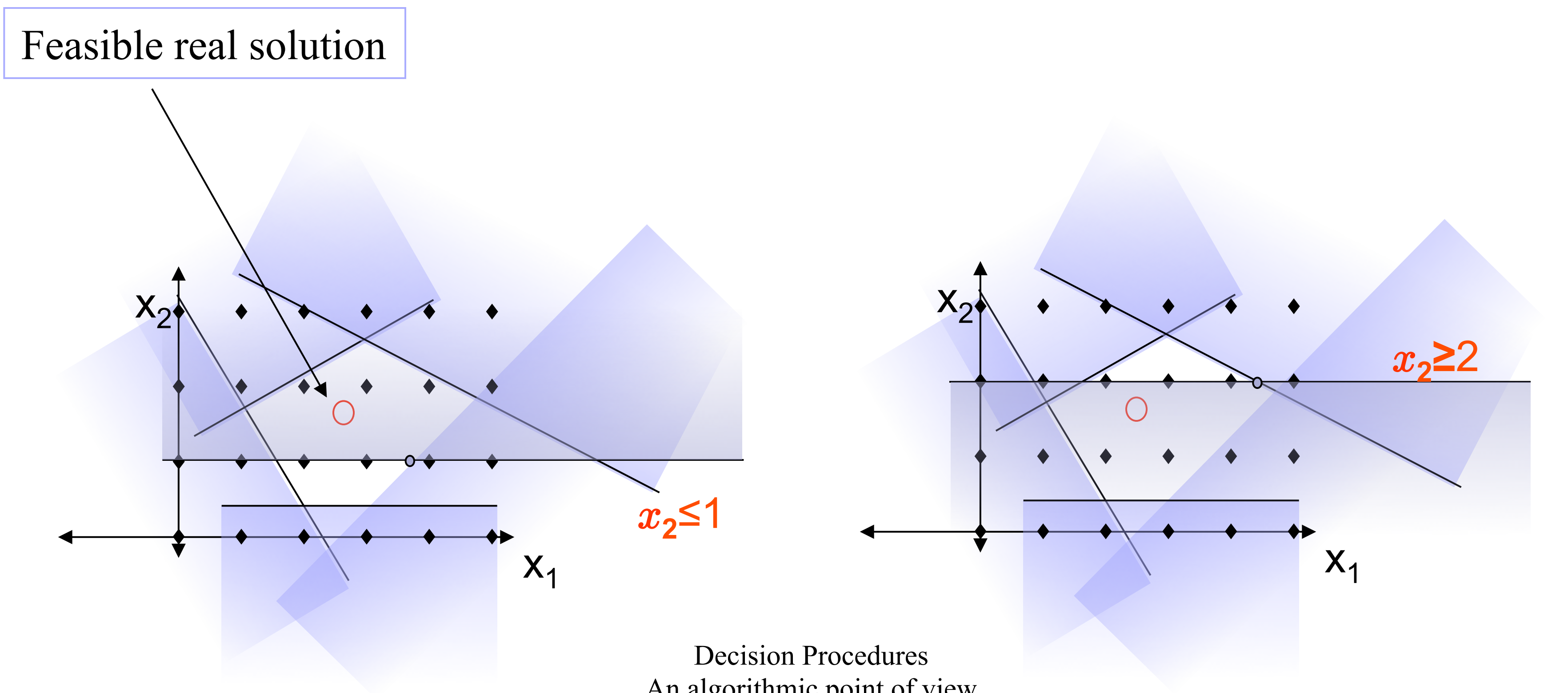
- Probably the most popular method for solving Integer Linear Programming (ILP) problems (First presented in 1960) is B & B.
- That is, the optimization problem.
- Recall, however, that we are interested in deciding feasibility of a linear system.
- In practice that's a little easier. The algorithm is quite similar.

Branch and Bound

- The main idea:
  - Solve the ‘relaxed’ problem, i.e. no integrality constraints.
  - If the relaxed problem is infeasible – backtrack (there is no integer solution in this branch)
  - If the solution is integral – terminate (‘feasible’).
  - Otherwise split on a variable for which the assignment is non-integral, and repeat for each case.
- More details to come…

Splitting on non-integral LP solutions.

- Solve LP Relaxation to get fractional solutions
- Create two sub-branches by adding constraints

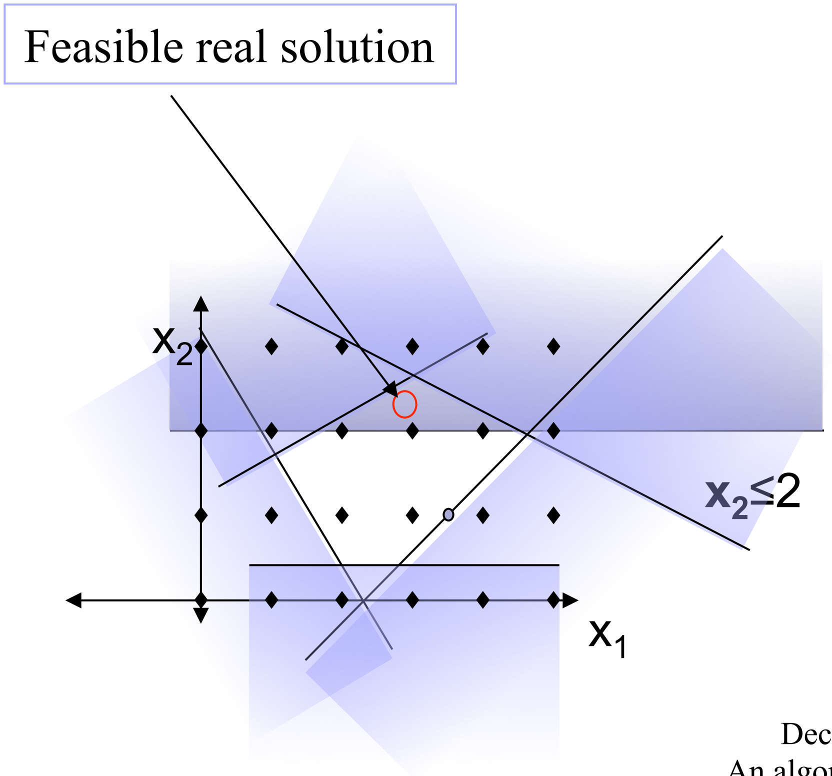


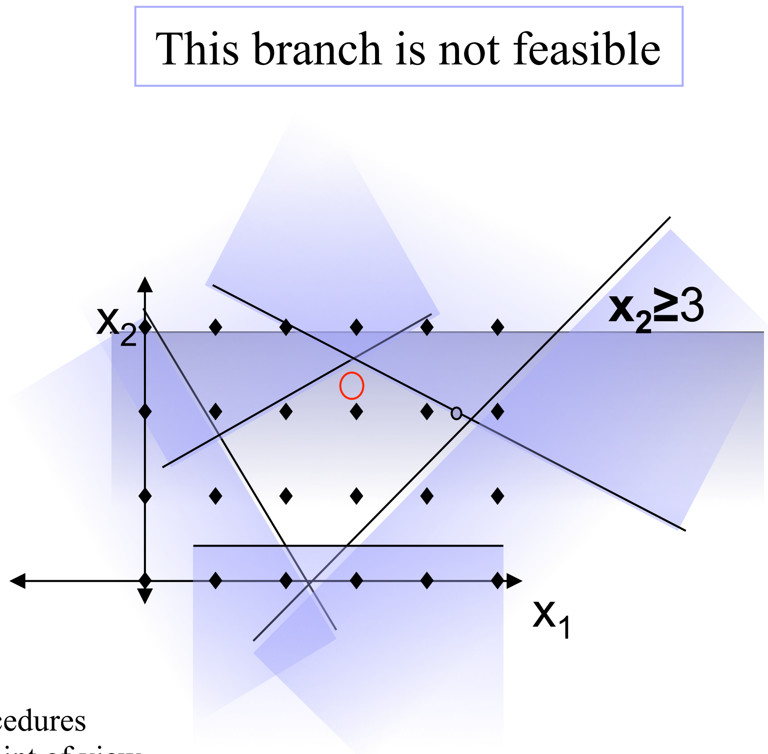
Example

- Suppose our system A has variables $x_{1\ldots}\ x_4$, and that the LP solver returned a solution (1, 0.7, 2.5, 3).
- Choose one of x_2, x_3. Suppose we choose x_2.
- Solve two new problems:
  - $A_1 = A \cup \{x_2 \leq 0\}$
  - $A_2 = A \cup \{x_2 \geq 1\}$
- Clearly A_1 or A_2 are satisfiable iff A is.

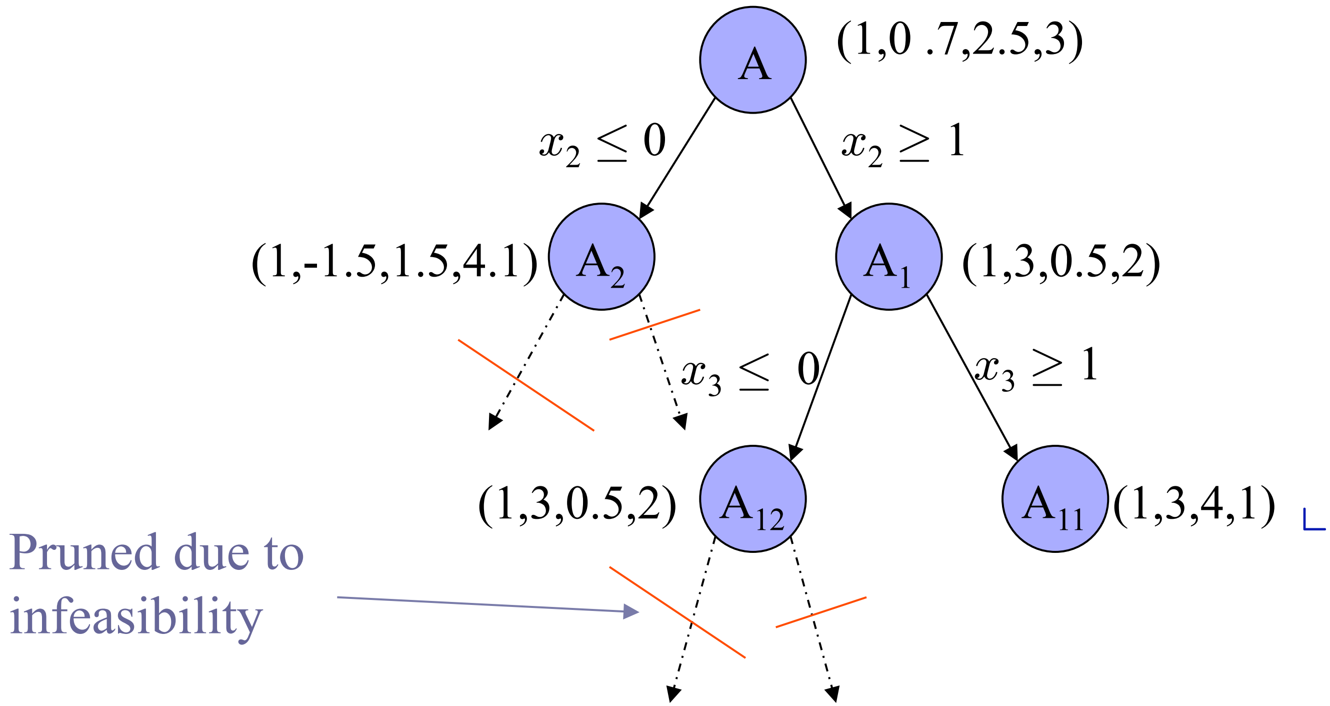
Splitting on non-integral LP solutions.

- The linear relaxation can also be infeasible…
- …which prunes the search for an integral solution.





The branch and bound tree



- Sub trees can be pruned away before reaching a leaf…
- Each leaf is a feasible solution.

Aside: B & B for optimality

- More reasons to prune the search.
- In a maximality problem:
  - Prune a branch if an over-approximation of the largest solution under this branch is still smaller than an under-approximation of the solution in another branch.
  - If the solution at the node is integral, update lower bound on the optimal solution, and backtrack.

Preprocessing (LP)…

- Constraints can be removed…
- Example:
  - $x_1 + x_2 \leq 2, \quad x_1 \leq 1, \quad x_2 \leq 1$
  - First constraint is redundant.
- In general, for a set:

$$S = \{x : a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b, l_j \leq x_j \leq u_j \text{ for } j = 0 \ldots n\}$$

$a_0 x_0 + \sum_{j=1}^{n} a_j x_j \leq b$ is redundant if $\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b$

Preprocessing (LP)…

- …and bounds can be tightened…
- Example:
  - $2x_1 + x_2 \leq 2, \quad x_2 \geq 4, x_1 \leq 3$
  - From 1st and 2nd constraints: $x_1 \leq -1$
- In general, if $a_0 > 0$

$$x_0 \leq (b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j)/a_0$$

- And, if $a_0 < 0$

$$x_0 \geq (b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j)/a_0$$

Preprocessing (ILP)

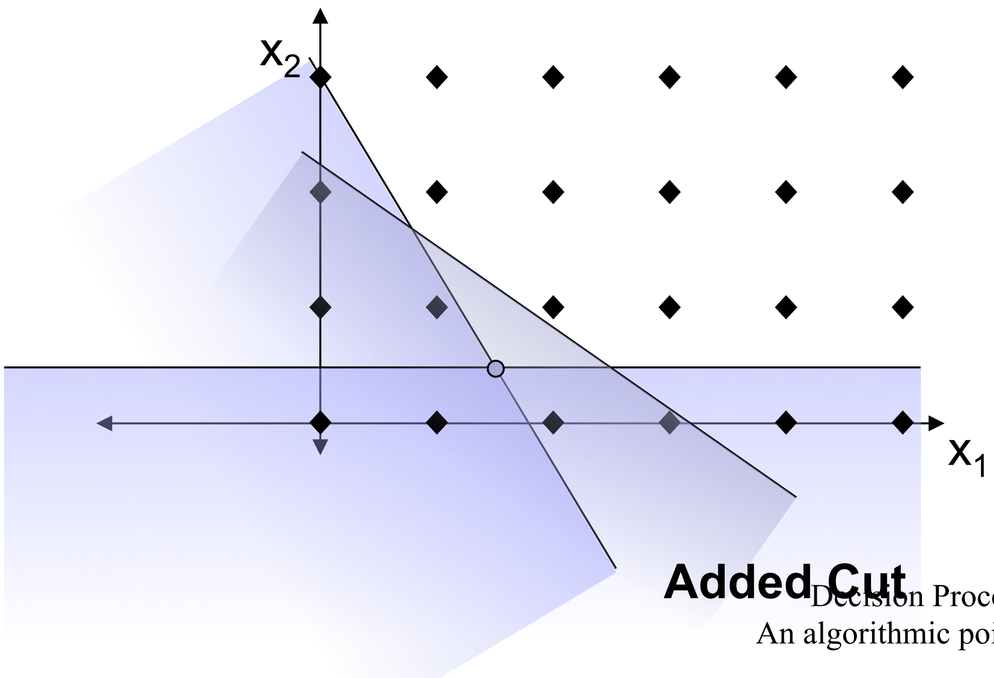
- Clearly $\lceil l_j \rceil \leq x_j \leq \lfloor u_j \rfloor$
- Consider a 0-1 ILP constraint: $5x_1 - 3\,x_2 \leq 4$
  - $x_1 = 1$ implies $x_2 = 1$
  - Hence, we can add $x_1 \leq x_2$
- (Again, a 0-1 ILP problem) Combine pairs:

  from $x_1 + x_2 \leq 1$ and $x_2 \geq 1$ conclude $x_1 = 0$;

- More simplifications and preprocessing is possible…
- The rule is: preprocess as long as it is cost-effective.

Improvement - Cutting Planes

- Eliminate non-integer solutions by adding constraints to LP (i.e. improve tightness of relaxation).



- All feasible integer solutions remain feasible
- Current LP solution is not feasible

Cutting planes

- Adding valid inequalities
- Examples:
  1. $x_1, x_2, x_3, x_4 \in \mathcal{B}$

     From $x_1 - x_2 + x_3 - x_4 \leq -1$

     … we can conclude $x_2 + x_4 \geq 1$

  2. $x \in \mathcal{Z}$

     From $2x \leq 11$

     …we can conclude $x \leq 5$