Integer Programming
ISE 418

Lecture 24

Dr. Ted Ralphs

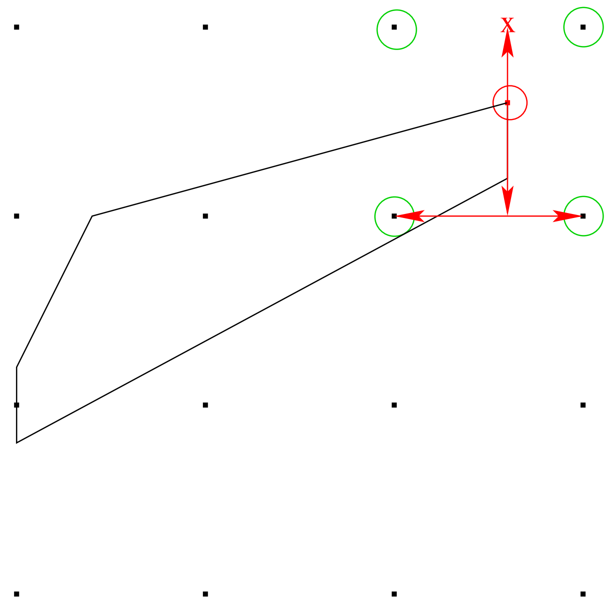
Reading for This Lecture

- “Primal Heuristics for Mixed Integer Programs,” Berthold

Heuristics in Integer Programming

- Heuristic methods are an extremely important aspect of integer programming in practice.
- Often it is the case that a near-optimal solution is "good enough."
- Furthermore, even if an optimal solution is required, heuristic methods can accelerate the solution process.
- Heuristic methods are generally used in one of two modes.
  - As a stand-alone procedure used directly to obtain a solution or as a means to obtain an initial bound (*metaheuristics*).
  - As an integrated part of a branch-and-bound procedure (*primal heuristics*).
- In this lecture, we will focus on the latter use, since this is the way in which heuristics are generally used in off-the-shelf solvers.

Simple Rounding

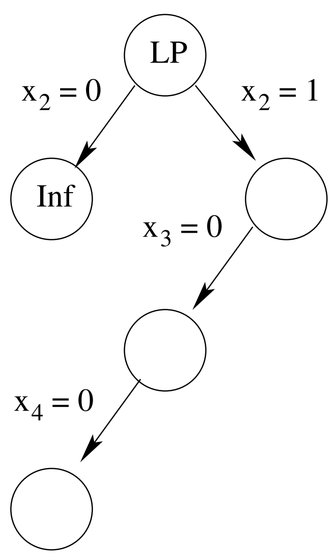


1. We first solve the LP relaxation to obtain an (infeasible) solution.
2. There may be a number of integer variables with fractional values.
3. We can round these variables one at a time, but there is no way to guarantee that this will lead to a feasible solution.
4. If there are k such variables, there are 2^k ways of rounding.
5. Use backtracking

Backtracking example

minimize x_1
subject to:
$x_1 - 2x_2 + 2x_3 + 2x_4 = -1$
$x_1 \geq 0$
$x_2, x_3, x_4 \in \{0, 1\}$

$\hat{x} = (0, 0.5, 0, 0)$



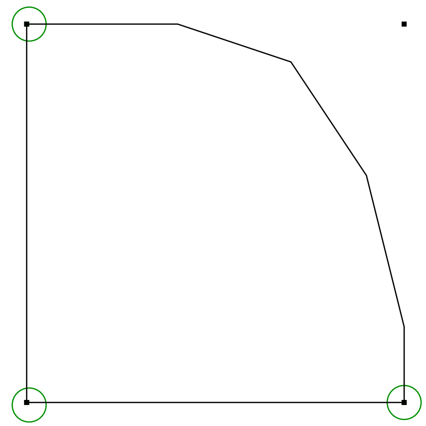
When an LP is solved after each fixing: Diving (Bixby et al., 2000)

Rounding

- Other variants
  1. Randomized rounding may help in specific contexts: single machine scheduling, set covering, set packing etc. (Bertsimas and Weismantel, 2005)
  2. Rounding problem can be explicitly stated as a binary program (Berthold 2006)
- Importance of rounding
  1. Rounding is cheap
  2. Many different variants of rounding may be deployed easily
  3. Rounding is an important step in several other heuristics

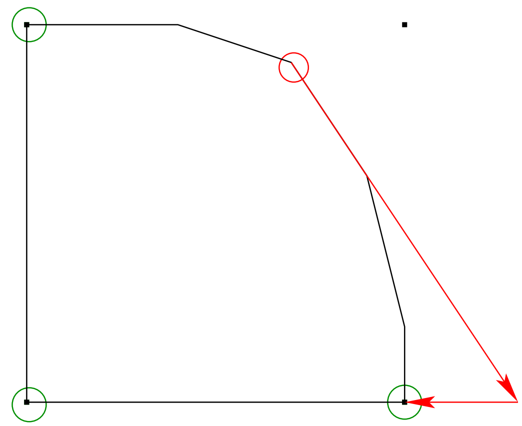
Simplex-based Heuristics

Many MILPs are MBPs $\Rightarrow$ Optimal solutions extreme points of the LP relaxation.



1. Pivot and complement (Balas and Martin, 1980)
2. Pivot cut and dive (Nediak and Eckstein, 2001)
3. OCTANE (Balas et al., 2001)
4. Pivot and shift (Balas, Schmieta, Wallace, 2004), for MILPs as well
5. Feasibility pump (Fischetti, Glover, Lodi, 2005)
6. Pivot and gomory cut (Ghosh and Hayward, 2006)

Pivoting schemes



- Explicit pivots to search neighboring basic feasible solutions with degraded objective values
- Pivots that move to an infeasible basic feasible solution (but with more variables at integer values)
- Pivots that restore feasibility
- Complementing variables to improve upon a known solution.
- Additional cuts
- Rounding

Feasibility Pump: The Basic Scheme

- We start from any $\hat{x}^0 \in \mathcal{P}$, and round to obtain $\widetilde{x}^0$.
- We look for a point $\hat{x}^1 \in \mathcal{P}$ which is *as close as possible* to $\widetilde{x}^0$ by solving the problem:

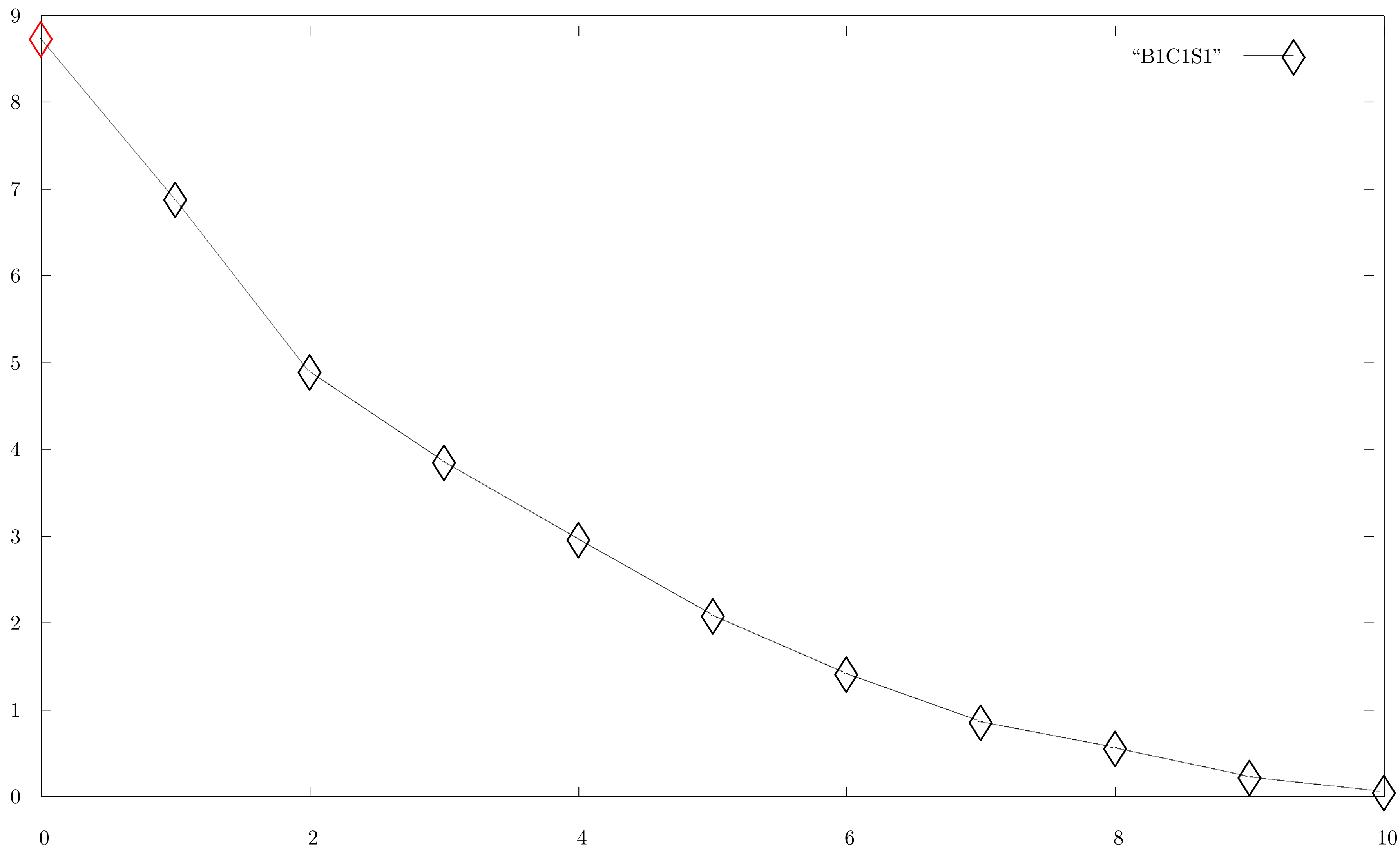
$$\min\{\Delta(x, \widetilde{x}) \mid x \in \mathcal{P}\}$$

  If we choose the measure $\Delta(x, \widetilde{x})$ properly, this problem is easily solvable.

- If $\hat{x}^1 \in \mathcal{S}$, we are done.
- Otherwise, we obtain $\widetilde{x}^1$ by rounding $\hat{x}^1$, and repeat.
- From a geometric point of view, this simple heuristic generates *two hopefully convergent trajectories of points $\hat{x}^i$ and $\widetilde{x}^i$*.
- These satisfy feasibility in a complementary but partial way:

1. $\hat{x}^i$, satisfies the linear constraints,
2. $\widetilde{x}^i$, the integrality requirements.

FP: Plot of the infeasibility measure $\Delta(\hat{x}^i, \widetilde{x}^i)$ at iteration i



FP: Definition of $\Delta(\hat{x}, \widetilde{x})$

- We consider the L_1*-norm distance* between a vector $x \in \mathcal{P}$ and a vector $\widetilde{x} \in \mathcal{S}$:

$$\Delta(x, \widetilde{x}) = \sum_{j \in I} |x_j - \widetilde{x}_j|$$

  where I is the set of indices of the integer variables.

- The *continuous variables* do not contribute to this function.

- In the case of a *binary* MILP:

$$\Delta(x, \widetilde{x}) := \sum_{j \in I : \widetilde{x}_j = 0} x_j + \sum_{j \in I : \widetilde{x}_j = 1} (1 - x_j)$$

- Given an integer $\widetilde{x}$, the *closest point* $\hat{x} \in \mathcal{P}$ can therefore be determined *by solving the LP*:

$$\min\{\Delta(x, \widetilde{x}) : Ax \leq b\}$$

FP: Implementation

- We can think of the distance as a pressure difference between $\hat{x}$ and $\widetilde{x}$ that we try to reduce by *pumping* the integrality of $\widetilde{x}$ into $\hat{x}$.
- On the other hand, it is clearly a measure of vicinity and therefore defines a neighborhood.
- The main problem with this method is stalling when $\Delta(\hat{x}, \widetilde{x})$ stops decreasing (we may produce the same solution).
  - In this case, we *reverse the rounding* of *some* variables $\hat{x}_j,\ j \in I$, even if this increases $\Delta(\hat{x}, \widetilde{x})$
  - This is done so as to minimize the increase in the current value of $\Delta(\hat{x}, \widetilde{x})$.

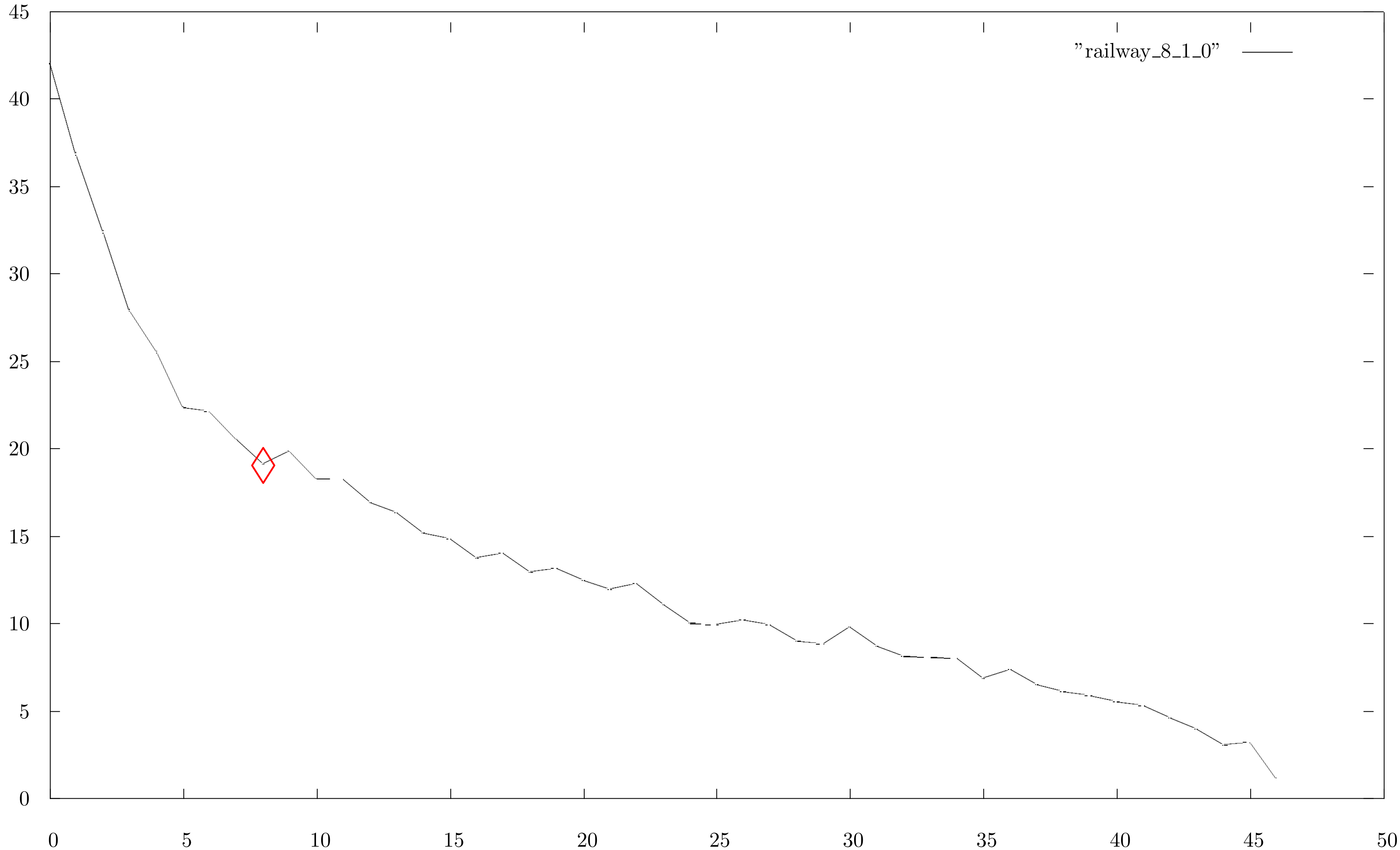
FP: A first implementation

1. initialize nIT := 0 and $\hat{x} :=$ argmax$\{c^\top x : Ax \le b\}$;
2. if $\hat{x}$ is integer, return($\hat{x}$);
3. let $\widetilde{x} := [\hat{x}]$ (= rounding of $\hat{x}$);
4. while (time < TL) do
5.   let nIT := nIT $+ 1$ and $\hat{x} :=$ argmin$\{\Delta(x, \widetilde{x}) : Ax \le b\}$;
6.   if $\hat{x}$ is integer, return($\hat{x}$);
7.   if $\exists\, j \in I : [\hat{x}_j] \neq \widetilde{x}_j$ then
8. $\widetilde{x} := [\hat{x}]$

        else

9.   flip the rand(T/2,3T/2) entries $\widetilde{x}_j$ with max $|\hat{x}_j - \widetilde{x}_j|$
10.   endif
11. enddo

FP: Plot of the infeasibility measure $\Delta(\hat{x}, \widetilde{x})$ at each pumping cycle



Neighborhood Search

- Rounding schemes explore neighborhoods defined by $\lfloor x_i^* \rfloor \leq x_i \leq \lceil x_i^* \rceil, i \in I$.
- Feasibility pump explores neighborhoods defined by the *nearby* basic feasible solutions
- Pivoting heuristics explores neighborhoods of $\hat{x}$ defined by the respective pivoting and complementing schemes.

Each of the above neighborhoods are explored using special methods

Exploring Neighborhoods

- The MILP solver itself can also be used as a search tool!
- A small neighborhood expressed as a MILP can be explored by using a MILP solver over it.
- Recall the "optimal rounding problem."

Local Branching

- Now assume we have a feasible solution $\bar{x}$, the so-called reference solution, and let $\overline{S} := \{j \in I \mid \bar{x}_j = 1\}$ denote the binary support of $\bar{x}$.
- For a given positive integer parameter k, we define the k-OPT neighborhood $\mathcal{N}(\bar{x}, k)$ of $\bar{x}$ as the set of the feasible solutions satisfying

$$\Delta(x, \bar{x}) := \sum_{j \in \overline{S}} (1 - x_j) + \sum_{j \in I \setminus \overline{S}} x_j \le k, \qquad (1)$$

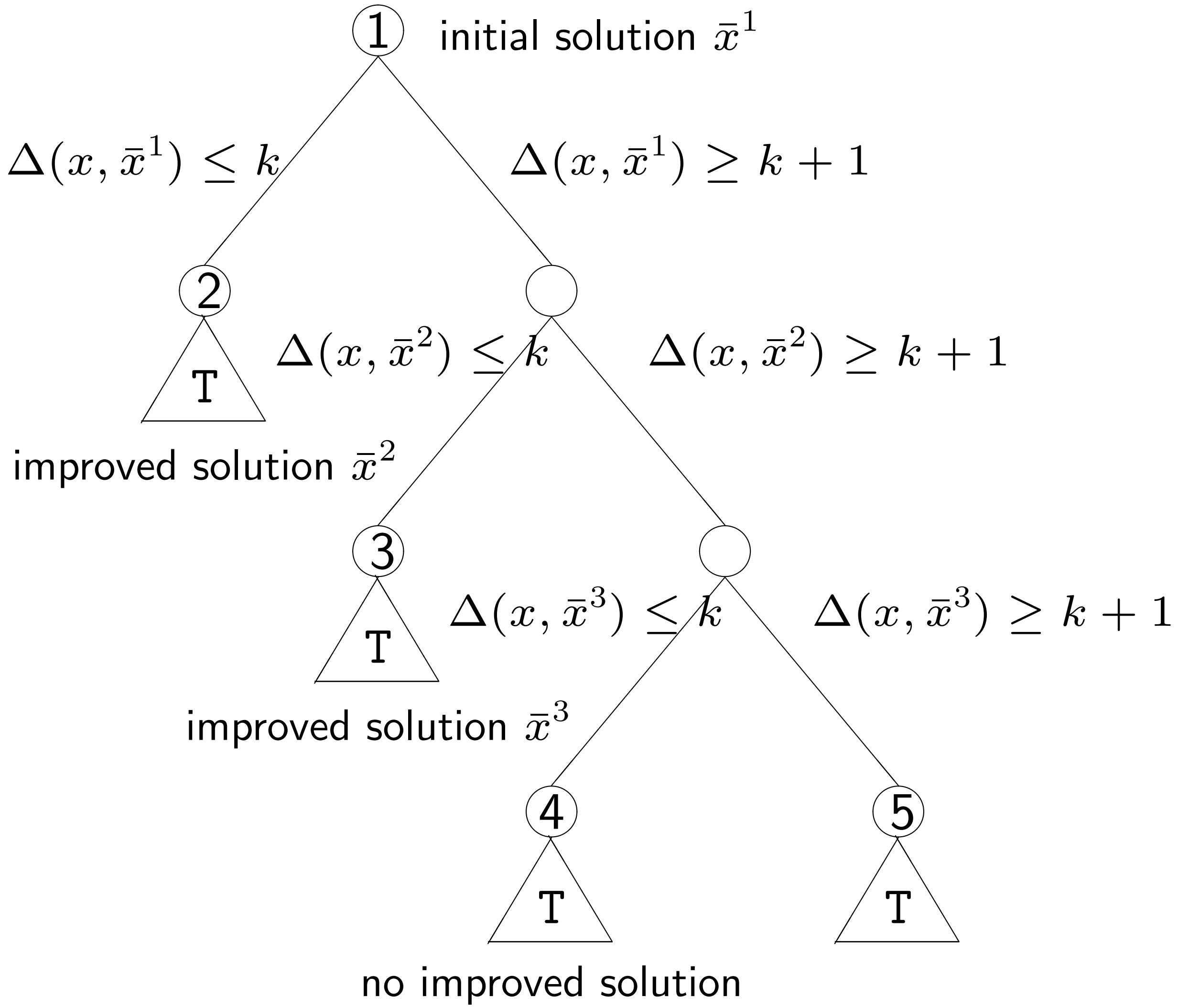
known as the *local branching constraint*.

- This constraint requires at most k variables have values different from $\bar{x}$.
- Constraint (1) can also be used to branch within a branch and bound:

$\Delta(x, \bar{x}) \le k$ (left branch) **or** $\Delta(x, \bar{x}) \ge k + 1$ (right branch)

- The neighborhoods defined by the local branching constraints can be searched by using a MILP solver recursively.

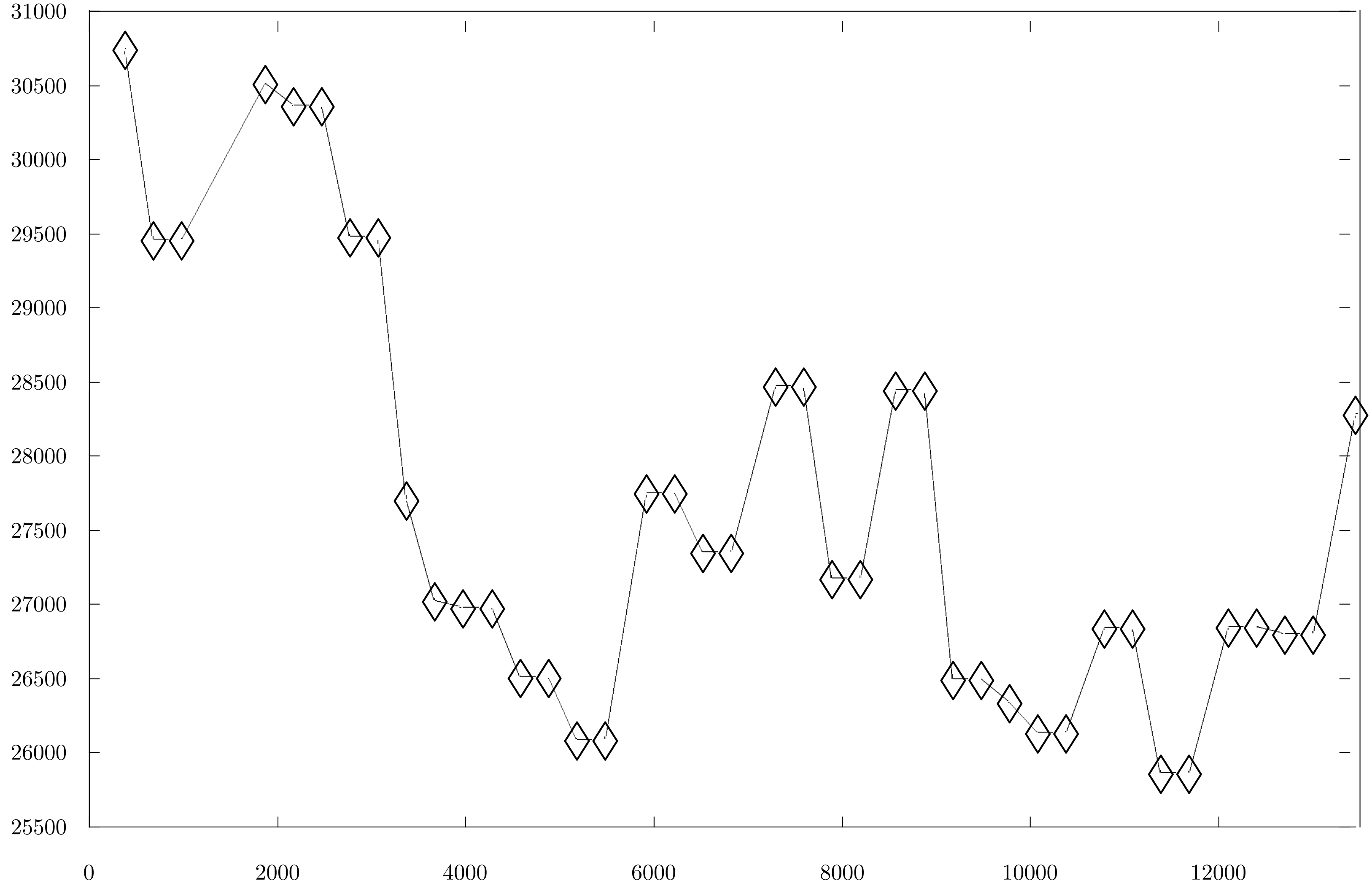
LB: The Basic Scheme



LB: Enhancements

- The previous scheme can be enhanced in two ways:
  - Imposing a time/node limit on the *left-branch* nodes:
    - In some cases, the exact solution of the left-branch node can be too time consuming for the value of the parameter k at hand.
    - Hence, from the point of view of a heuristic, it is reasonable to impose a time/node limit for the left-branch computation.
  - Increasing diversification:
    - A further improvement of the heuristic performance can be obtained by exploiting diversification mechanisms in the spirit of metaheuristic techniques.
    - In this scheme, diversification is applied by *varying the value of* k and accepting non-improving solutions.
- On the other hand, it is easy to see that an alternative implementation would be within the branch-and-cut tree of a MILP solver.
- More precisely, we search using the branch-and-cut algorithm itself for a *fixed number of nodes*.
- Whenever a new incumbent has been found, this LB can be fed into this local search to limit enumeration.

LB: solution value vs. CPU seconds for instance B1C1S1



Relaxation Induced Neighborhood Search

- A similar concept of neighborhood takes into account simultaneously both
  - the *incumbent* solution $\bar{x}$, and
  - the *the solution of the continuous relaxation* $\hat{x}$,

  at a given node of the branch-and-bound tree.
- $\bar{x}$ and $\hat{x}$ are compared and all the binary variables that assume the same value are *hard-fixed* in an associated MILP.
- This associated MILP is then solved by using the MILP solver as a black-box.
- In case the incumbent solution is improved, $\bar{x}$ is updated in the rest of the tree.
- This method turns out to give very competitive results on general MILPs.
- It is particularly suitable in the scheduling context where the problem is very constrained and any non-trivial value of k would be too large.

RINS Formulation

RINS: Let $\tilde{x}$ be a known feasible solution and let $\hat{x}$ be an LP-solution at some node in the search tree. We create a new MILP (Danna et al., 2005):

minimize $z = cx$

s.t.

$$Ax \leq b,$$

$$x_i = \tilde{x}_i \quad \forall i \quad \text{s.t. } \tilde{x}_i = \hat{x}_i$$

$$x \in \mathbb{Z}^r \times \mathbb{R}^{n-r}.$$

Working with Infeasible Solutions

- Sometimes waiting to have a fully feasible solution before starting a local search approach is unnecessary.
- Combining the work of both *FP and LB* provides the following more flexible scheme:
  1. FP is executed for a *limited number of iterations* and the *integer (infeasible)* solution $\widetilde{x}$ with minimum distance Δ to a feasible solution $\hat{x}$ of the LP relaxation is stored;
  2. LB starts by using $\widetilde{x}$ *as a reference solution*, replacing the original objective function with

$$\min \sum_{i \in T} y_i$$

     where T is the set of the indices of the constraints violated by $\widetilde{x}$ and a binary variable y_i has been defined for each constraint $i \in T$.

Working with Infeasible Solutions (cont.)

- Hence, in a first phase, LB attempts to improve the current infeasible solution by reducing the number of infeasible constraints in the spirit of the first phase of the simplex algorithm.
- In the second phase, once a feasible solution has been found, the original objective function is then restored and LB takes care of *improving the quality* of such a solution.