Is more information better?*

Rodolfo Carvajal
ISyE, Georgia Institute of Technology

September 22, 2013

Abstract

We perform a simple computational experiment to test the effect of providing a state-of-the-art commercial solver with the optimal solution *a priori*.

1 Introduction

A while ago I experienced a behaviour in CPLEX that I thought it was strange, or at least counterintuitive: Solving a Mixed Integer Linear Problem (MILP) when knowing in advance the optimal solution would in cases be (a lot) harder than just starting from scratch. How could it be that having more information would hurt the performance of a MILP solver? So, I decided to perform a small computational study[1], exploring the effect of providing primal information to a MIP solver in its performance.

2 The experiment

Given a MILP instance:

1. We solve it for five hours (wall-clock time) with a MILP solver.
2. Solve the instance for five hours again, but this time giving the solution found in 1. as a MIP start to the MILP solver.

*Originally posted on December 24, 2012 at `http://rocarvaj.calepin.co`.
[1]This work is part of ongoing research with G. Nemhauser and S. Ahmed.

3 Experimental setting

CPLEX 12.4[2] (with 8 threads) was used as the MILP solver. We use instances from the *Benchmark* class in MIPLIB 2010[3], removing infeasible instances. Instance `bnatt350.mps.gz` was removed because CPLEX could not find any feasible solutions within the five-hour limit. This leaves us with 83 instances.

4 Results

Out of the 83 instances, two of them were not solved to optimality in none of the two runs (`n3div36.mps.gz` and `m100n500k4r1.mps.gz`), and the instance `n3seq24.mps.gz` was not solved to optimality by CPLEX + MIP Start.

Table 1 presents geometric means for time and number of branch-and-bound nodes required by each configuration (Default CPLEX and Default CPLEX + MIP Start) to prove optimality, computed over the 81 instances solved to optimality by both configurations. We have divided the instances in three classes, according to the maximum number of branch-and-bound nodes that it took for the two configurations to solve to optimality: 0 to 5999 nodes (class 1), 6000 to 59999 nodes class 2, and 60000 nodes and up (class 3). The row *improvement* indicates the relative improvement (or detriment) of the geometric mean with respect to Default CPLEX.

As we can see, the effect of providing this type of primal information to the solver is valuable in terms of time and branch-and-bound nodes, consistently for the three instance classes. The benefits of providing the information seem to decrease when the difficulty of the instances increases (from class 1 to class 3).

4.1 But is not all good...

An important observation is that there exists instances in which providing the optimal solution as information *a priori* can actually be harmful for the performance of the solver. Two notorious examples are:

- Instance `n3seq24.mps.gz`, solved by default CPLEX in 876.80 seconds and exploring 35999 nodes, but proved to be within 0.38% of the optimal value by default CPLEX + MIP Start in 18000 seconds and exploring 616345 nodes.
- Instance `enlight13.mps.gz`, solved by default CPLEX in 57.53 seconds and exploring 54753 nodes, but solved by default CPLEX + MIP Start in 2457.11 seconds and exploring 1933920 nodes. This represents a *detriment* of 4171.01% in time and 3432.02% in nodes!

[2] `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer`

[3] Koch, Thorsten, et al. "MIPLIB 2010." *Mathematical Programming Computation* 3.2 (2011): 103-163.

Class	Configuration	Time	Nodes
1	Default CPLEX	126.16	251.94
	Default CPLEX + MIP Start	34.57	87.37
	Improvement	72.60%	65.32%
2	Default CPLEX	226.71	16170.31
	Default CPLEX + MIP Start	82.63	8632.35
	Improvement	63.55%	46.62%
3	Default CPLEX	984.86	269510.48
	Default CPLEX + MIP Start	398.80	175403.25
	Improvement	59.51%	34.92%

Table 1: Geometric means for time and number of branch-and-bounds nodes required to prove optimality.

To get an idea of how bad this is in the case of `enlight13.mps.gz`, below are plots for the behaviour of the best feasible solution found and best lower bound, in terms of the number of branch-and-bound nodes explored, for both Default CPLEX and Default CPLEX + MIP Start.

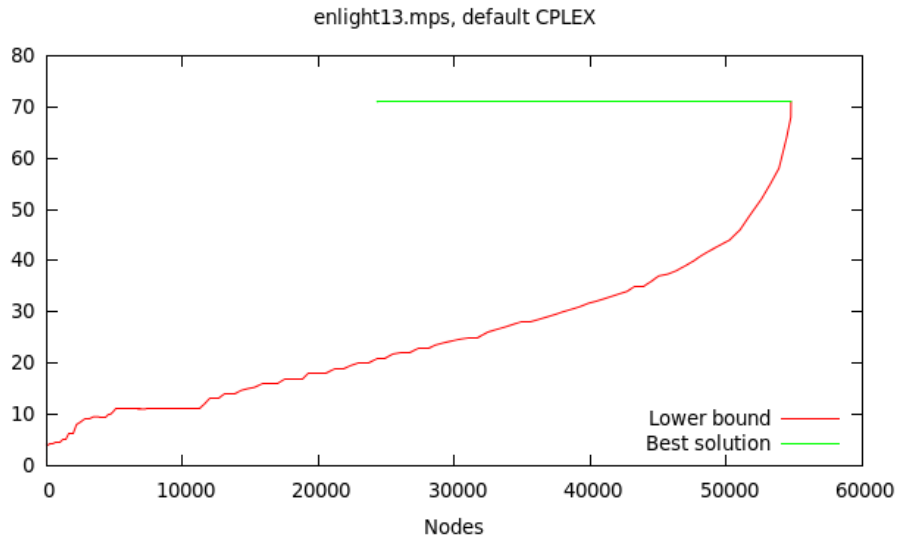

Figure 1: Default CPLEX

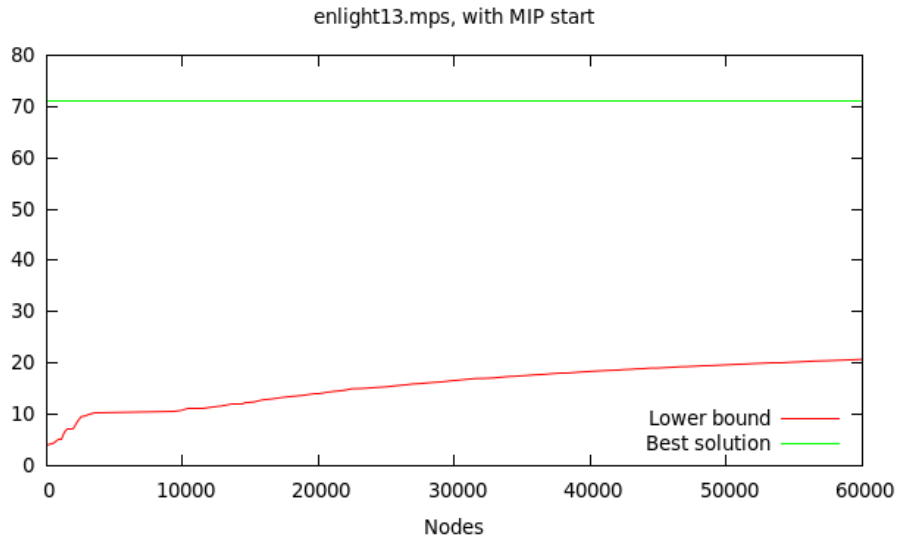

Figure 2: Default CPLEX + MIP Start

One argument that could be made is that in the case where CPLEX knows the optimal solution in advance, it expends a lot of time in heiristics that do not report any benefits. This could explain the detriment in time, but not in branch-and-bound nodes. Somehow, knowing the optimal solution changes the search tree in very mean ways.

Edit: Jean Francois Puget posted a nice response to this article here.