Project 2
Branch and Bound solutions for MILPs

Matt Kingsinger, Hongjun Choi, Adarsh Akkshai

September 13, 2017

1. (a) We need to convert sample submission in NEOS-MILP-MOSEK solver, **HDRXMIP1.MPS**, into LP format. Below is the converted LP format that will be suitable for submission to the SCIP solver:

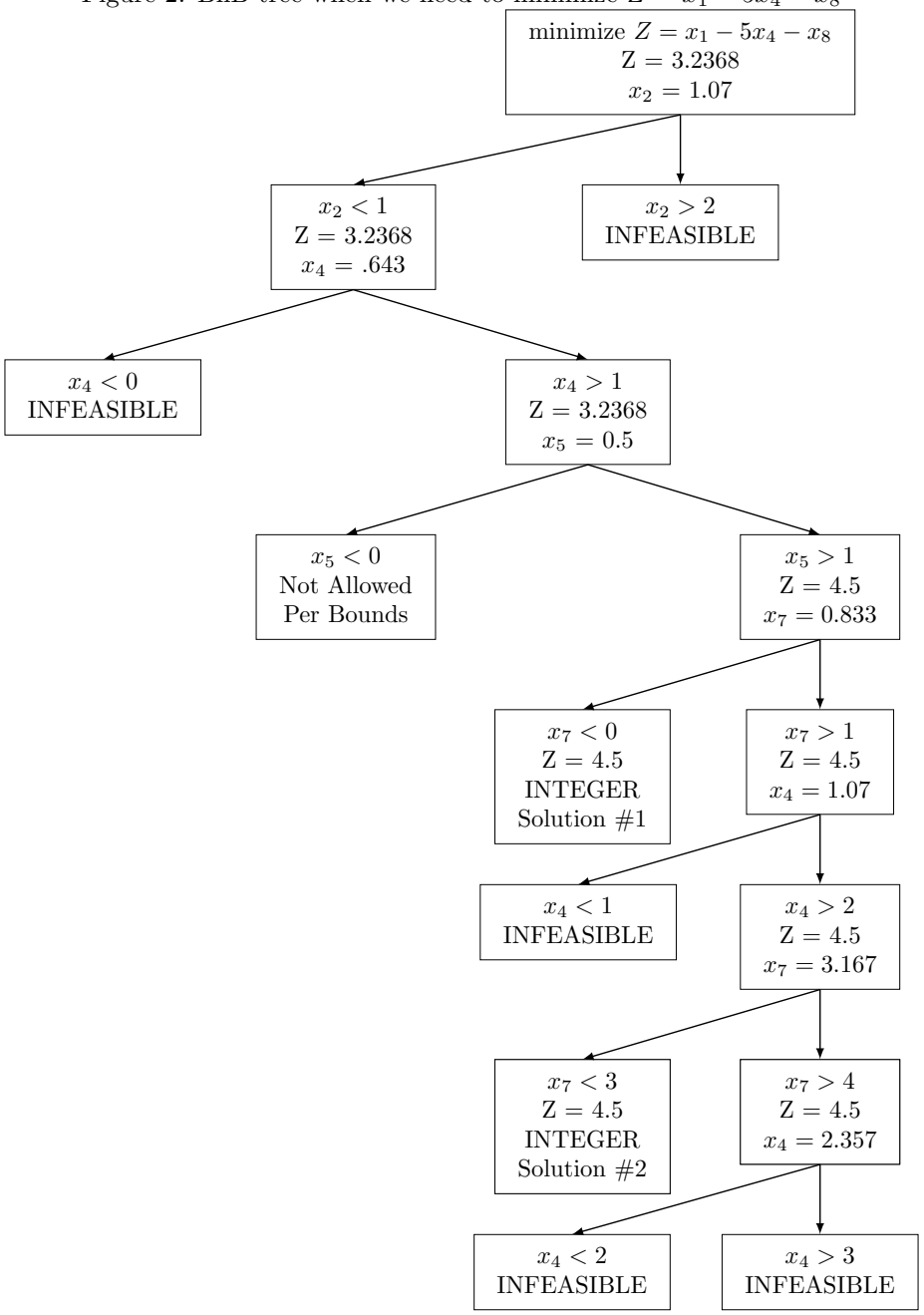
Figure 1: The problem in LP format

```
minimize
 obj: x1 + 2 x5 - x8
subject to
 c1:   3 x1 +   x2         -  2 x4    - x5                  -    x8 >= 2.5
 c2:          2 x2 + 1.1 x3                                      <=  2.1
 c3:                      x3                 +  x6                =  4.0
 c4:                         2.8 x4               -1.2 x7          >=  1.8
 c5:                         2.8 x4               -1.2 x7          <=  5.0
 c6: 5.6 x1                             + x5               + 1.9 x8 >=  3.0
 c7: 5.6 x1                             + x5               + 1.9 x8 <= 15.0
bounds  \default bounds are 0 <= x < +inf
 2.5 <= x1 < +inf
   0 <= x2 <= 4.1
 0.5 <= x5 <= 4.0
   0 <= x6 < +inf
   0 <= x7 < +inf
   0 <= x8 <= 4.3
end
```

We will need to solve this problem as a MIP where we consider all variables to be integers **except** x_1. To do this we will perform Branch and Bound (BnB) manually, and then confirm our results by placing integer constraints into our .lp file and having the CPLEX solver find the MIP solution for us.

On the next page is the layout of our manual branch and bound process. We performed a depth-first search and found solution #1. Then we completed all of the branches to confirm that solution #1 was the optimal solution.

Figure 2: BnB tree when we need to minimize $Z = x_1 - 5x_4 - x_8$



INTEGER Solution #1	$x_1 = 2.5$
Objective = 4.5	$x_2 = 0$
	$x_3 = 0$
	$x_4 = 1$
	$x_5 = 1$
	$x_6 = 4$
	$x_7 = 0$
	$x_8 = 0$

INTEGER Solution #2	$x_1 = 2.5$
Objective = 4.5	$x_2 = 0$
	$x_3 = 0$
	$x_4 = 2$
	$x_5 = 1$
	$x_6 = 4$
	$x_7 = 3$
	$x_8 = 0$

Table 1: There are two solutions when we use BnB for problem 1 (a)

We see that we have found two solutions with the same objective value. There is not a unique solution to this MIP (see **HDRXMIP1.lp** for the code that includes all of our branch and bound iterations).

1.(b) We will now confirm the solution we found by including the integer constraints in the .lp file and submitting it to SCIP to solve (see **HDRX_int.lp and NEOS-MILP-SCIP-CPLEX_integer.txt** for the LP file and the SCIP output):

```
minimize
 obj: x1 + 2 x5 - x8
subject to
 c1:   3 x1 +   x2          -  2 x4    - x5                  -    x8 >= 2.5
 c2:          2 x2 + 1.1 x3                                        <=  2.1
 c3:                      x3                +  x6                  =  4.0
 c4:                         2.8 x4               -1.2 x7          >=  1.8
 c5:                         2.8 x4               -1.2 x7          <=  5.0
 c6: 5.6 x1                              + x5               + 1.9 x8 >=  3.0
 c7: 5.6 x1                              + x5               + 1.9 x8 <= 15.0
bounds  \default bounds are 0 <= x < +inf
 2.5 <= x1 < +inf
   0 <= x2 <= 4.1
 0.5 <= x5 <= 4.0
   0 <= x6 < +inf
   0 <= x7 < +inf
   0 <= x8 <= 4.3
general
x2 x3 x4_x5 x6 x7 x8
end
```

Figure 3: The problem when we restrict x_2 through x_8 to integer values

$$\textbf{(x1,x2,x3,x4,x5,x6,x7,x8) = (2.5,1,0,1,1,4,0,0)}$$

with objectve = 4.5. Hence the solution that we found using branch and bound was correct.

1. (c) Consider the binary LP (see **prob1c.mod** for AMPL model):

$$\begin{aligned} &\textit{minimize} && x_{n+1} \\ &\textit{subject to} && 2\sum_{i=1}^{n} x_i + x_{n+1} \quad = 2k+1, \quad x \in \{0,1\}, k \in \mathbb{Z}, k \leq n. \end{aligned}$$

Since, the constraint is equality with an odd number and all x_i, $i \neq n+1$ are summed and then multiplied by 2, the only way to get satisfy the constraint with all x_i being binary is if $x_{n+1} = 1$.

While performing BnB we will try to begin with the constraints that all $0 \leq x_i \leq 1$. The solution will give us a minimized objective of $x_{n+1} = 0$, and some $x_i = 0.5$. We will then branch on this x_i and force it to be binary, causing two branches $x_i = 0$ or $x_i = 1$ (assuming feasibility) the solver will choose some other $x_j = 0.5$, where ,$j \neq i$. During the next depth node we re-iterate this process and branch on x_j. Every new branch that we make will lead to two new branches until we have either

i. $x_i = 1$ for k i's

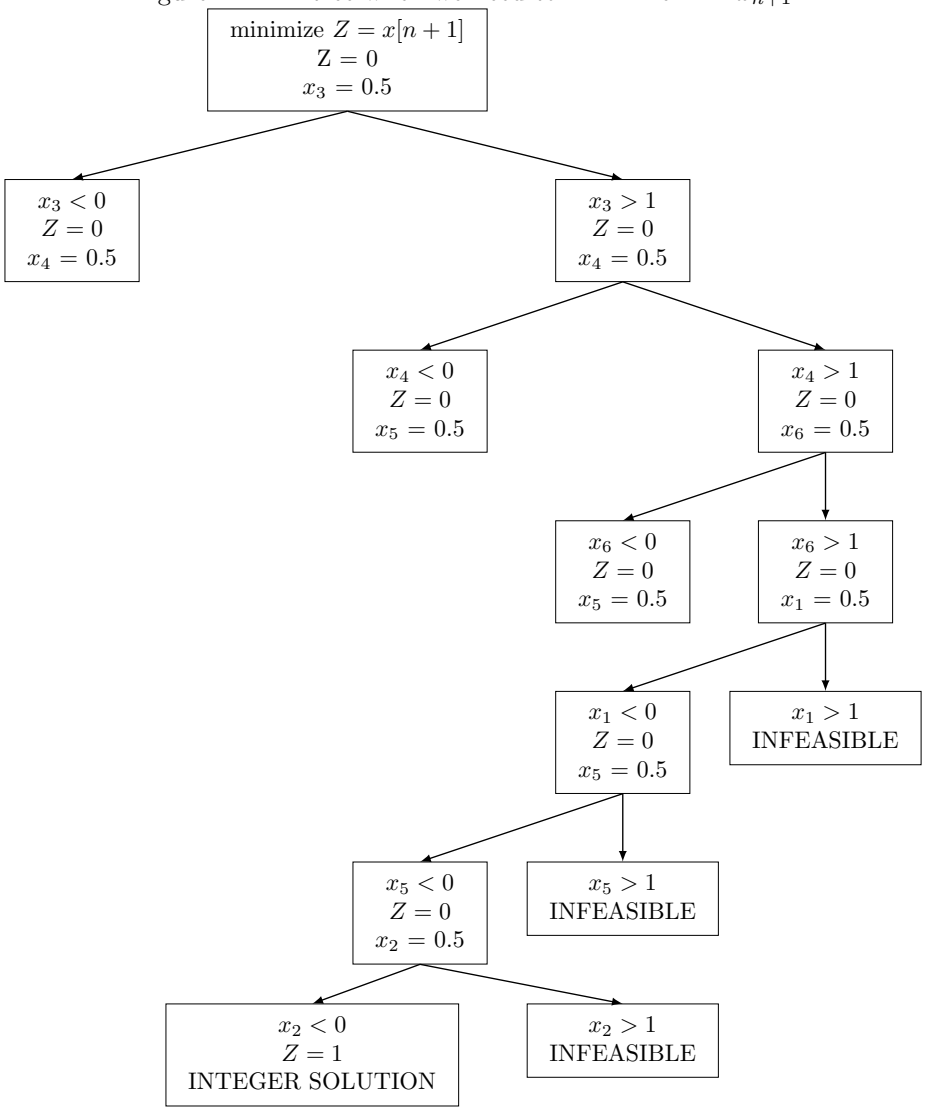
ii. $x_i = 0$ for n-k i's

At this point there will still be one feasible branch pertaining to $x_i = 0$ or $x_i = 1$, respectively. We will then be forced to continue solving iterations with a single branch all the way until we have branched on every x_i, $i = \{1, 2, \ldots, n\}$. This is because of the symmetry of the problem. When you isolate one variable to be either 1 or 0, the solver will simple move on to another variable that is still free to be set to optimally solve the problem, in our case 0.5. Thus any new feasible branch created at level p, where $p \leq n$, will need to be solved $n-p$ more times. Moreover, at any node where this branch leads to two feasible solutions, we will have created a whole new branch that also needs to be solved through to the n^{th} variable. This leads to an explosive growth in computation time for n large, and it gets even worse for k which is near to $\frac{n}{2}$. Having k near to zero or near to n (always $k \leq n$) will decrease the number of new feasible branches at levels deeper than $\min\{k, n-k\}$. We will see a kind of symmetry in our branch and bound tree where we have sides of length k and $n-k$. Hence when $k = n-k$, i.e. $k = \frac{n}{2}$ we will see the largest increase in the number of nodes computed for each increment of n. Even in cases where $k = 0$ or $k = n-1$ we would still need to follow a single branch all the way through all x_i, $\{i = 1, 2, ..n\}$.

As an example we will perform a depth first manual branch and bound on a problem with $n = 6$ and $k = 3$. This will produce a single sequence of branches that needs to be solved to n^{th} level. We will not complete all other branches of the tree, but you will see that there are many other branches that would need to be solved through the n^{th} variable (see **prob1c.lp** and **NEOS_CPLEX_prob1c2_output.txt** for the .lp file and SCIP output).

The next page has the BnB tree done manually.

Figure 4: BnB tree when we need to minimize $Z = x_{n+1}$



(Note that the depth-first search has gone down $n = 6$ levels)

To avoid manual labor we restrict $k = 8$ and n $= \{7, \ldots, 30\}$. We will now use the NEOS-CPLEX and NEOS-CBC solvers to understand the problem of symmetry and the growth of the complexity. Since the solvers have defaults set to solve LP problem with cuts, pre-solve and heuristics this would circumvent the problems caused by symmetry/ambiguity in the problem statement. Hence we manually enforce all cuts to 'false' and disable all presolve and heuristic capabilities to see the full affect of the symmetry of the problem on the BnB process.

Table 2: The list of commands to be turned off for BnB problem 1 (c) in CPLEX, (see **opt_cplx.txt** for CPLEX commands file)

Options	Description
'cutpass=-1';	#sets the number of cuts to zero
'coeffreduce=0';	#do not use coefficient reduction when preprocessing MIPS
'cutstats=1';	#display the number and kinds of cuts performed (should be no cuts)
'disjcuts=-1';	#do not generate MIP disjunctive cuts
'flowcuts=-1';	#do not use flow cuts when solving MIPs
'flowpathcuts=-1';	#do not generate MIP flow-path cuts
'fraccand=0';	#Limit on candidate variables for Gomory cuts in MIP problems
'fraccuts=-1';	#do not generate MIP fractional Gomory cuts
'gubcuts=-1';	#do not use GUB cuts in solving MIPs
'impliedcuts=-1';	#do not use implied cuts in solving MIPs
'localimpliedcuts=-1';	#do not generate locally valid implied bound cuts for MIPs
'mcfcuts=-1';	#do not use multi-commodity flow cuts
'cliques=-1';	#do not use clique cuts in solving MIPs
'covers=-1';	#do not use cover cuts in MIPs
'mipcuts=-1';	#turns off all cuts
'mipdisplay=1';	#display branch-and-bound information at each integer feasible solution
'mipsearch=1';	#traditional branch-and-cut strategy for MIPs
'mipstartvalue=0';	#do not use initial guesses in problems with integer variables
'mircuts=-1';	#do not generate MIP rounding cuts
'nodesel=0';	#depth first branch and bound
'nodesel=1';	#breadth first branch and bound
'prepass=0';	#limit on number of CPLEX presolve passes
'prerelax=0';	#do not use CPLEXs presolve on the initial LP relaxation of a MIP
'presolve=0';	#do not run CPLEXs presolve algorithm
'presolvenode=-1';	#do not run presolve at each node of the branch and bound tree
'priorities=0';	#do not consider priorities for MIP branching
'probe=-1';	#do not do variable probing when solving MIPs
'repeatpresolve=0';	#do not repeat CPLEX's presolve at MIP nodes
'splitcuts=-1';	#do not use lift-and-project cuts on MIPs
'symmetry=0';	#do not break symmetry during preprocessing of MIPs
'varsel=-1';	#branch on variable with smallest integer infeasibility
'zerohalfcuts=-1';	#do not generate zero-half cuts for MIPs

Table 3: Total nodes traversed for CPLEX solver, k = 8

n	**Number of nodes in Branch and Bound**
7	Infeasible
8	Presolved, no nodes
9	0
10	54
11	219
12	714
13	2001
14	5004
15	11439
16	24309
17	48619
18	92377
19	167959
20	293929
21	497419
22	817189
23	1307503
24	2042974
25	3124549
26	4686824
27	6906899
28	10015004
29	14307149
30	20160074

(We were unable to display computational time for the CPLEX solver)

We repeat this process with the CBC solver,

Options	Description
'cuts=Off'	Turn off all cuts
'preprocess=Off	Turn off all pre-solving for the LP
'heuristics=Off'	Turn off any heuristics for BnB tree

Table 4: Options to turn off for the CBC solver, (see **opt_cbc.txt** for CBC solver options file)

Table 5: k = 8, CBC solver and total nodes traversed

n	**Number of nodes in Branch and Bound**	computational time
7	Infeasible	n/a
8	Presolved	n/a
9	0	0.002
10	24	0.004
11	128	0.012
12	412	0.037
13	1124	0.078
14	5360	0.153
15	18230	0.296
16	44928	0.464
17	95096	0.722
18	183804	1.372
19	335218	2.22
20	586594	3.81
21	993082	5.99
22	1632008	10.58
23	2612492	16.82
24	4082726	26.03
25	6241158	41.77
26	9178530	60.18
27	13402524	91.69
28	19773624	119.11
29	28332278	174.58
30	39754580	245.24

As n approached 30, even for problems submitted locally to the server, we were experiencing computation times of five to ten minutes. We can see in different sections of the table the number of nodes computed in the BnB process increases by a factor of anywhere between 5 and $\frac{3}{2}$ each time n increases by 1 (this is a rough approximation). We can summarize our observations in each table as:

n relative to k	Rough approximate of factor α of increase in # of nodes per increment in n
$k << n$	$3 \leq \alpha \leq 5$
$k \approx \frac{n}{2}$	$\alpha \approx 2$
$k \to n_-$	$\alpha \approx \frac{3}{2}$

This table highlights the detrimental growth of the computational time for MIPs with symmetry when utilizing solely Branch and Bound to find the solution.