Boolean LP

December 4, 2021

[1]:
```
import cvxpy as cp
import numpy
```

[2]:
```
numpy.random.seed(0)  # set seeds
n = 100
m = 300
# generate random data for the relaxed LP
A = numpy.random.uniform(size=(m, n))
b = A @ numpy.ones(n) / 2
c = -numpy.random.uniform(size=n)
# souling the relaxed problem
x = cp.Variable(n)
objective = cp.Minimize(c @ x)
constraints = [A @ x <= b, x >= 0, x <= 1]
prob = cp.Problem(objective, constraints)
prob.solve()
print("status:", prob.status)
print("optimal value", prob.value)
```

```
status: optimal
optimal value -34.41722425996274
```

Now we shall find

$$\hat{x} = \begin{cases} 1 & x_i^{rlx} \geq t \\ 0 & otherwise \end{cases}$$

Note that $c^T\hat{x}$ is an upper bound on the $c^T x^*$ of the Boolean LP, while $c^T x^{rlx}$ is a lower bound on $c^T x^*$.

We take 100 values of t sampled uniformly from $[0, 1]$, and calculate the smallest difference $c^T(\hat{x} - x^{rlx})$. This gives an indication of how good our approximaiton was. [of course only for feasible $\hat{x}$, i.e. $A\hat{x} \leq b$]

[3]:
```
tspan = numpy.linspace(0, 1, 100)
max_error = numpy.zeros(100)
est = numpy.zeros(100)

for i in range(100):
    x_hat = numpy.array([int(e >= tspan[i]) for e in x.value])
    max_error[i] = numpy.max(A @ x_hat - b)
```

```
    est[i] = c @ x_hat

min(est[max_error <= 0]) - prob.value
```

[3]: 0.8399729146557178