**Current methods to apply cardinality constraint to my problem:**

* Apply an inverse OR penalty to each group of 7 variables. Then take the resulting value away from K (the cardinality constraint).
* Link every 7 variables to 1 slack variable. Then apply a penalty in the sum of the slack variables is not K.

**Apply an OR penalty to each group of 7 variables**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Easy to accomplish | Links lots of variables together. Which is bad |
| Works without introducing extra variables | Final model is hard to read |

I decided to go with an inverse OR penalty here because it makes the culmination of the penalty clearer. Initially I was going to make it a regular OR but after running into problems where the nested penalties had to be inverse to work correctly it made more sense to just use all inverse OR penalties.

Example of a nested inverse OR penalty for 3 variables, produced by my code:

((Binary('x[0]') + ((Binary('x[1]') + Binary('x[2]')) + (-1.000000 \* (Binary('x[1]') \* Binary('x[2]'))))) + (-1.000000 \* (Binary('x[0]') \* ((Binary('x[1]') + Binary('x[2]')) + (-1.000000 \* (Binary('x[1]') \* Binary('x[2]')))))))

**This was incorrect as the OR penalties needed to be inversed so that they acted like regular binary variables.**

Example of a nested OR penalty for 3 variables, produced by my code:

(((1.000000 + (-1.000000 \* Binary('x[0]'))) + (-1.000000 \* (((1.000000 + (-1.000000 \* Binary('x[1]'))) +

(-1.000000 \* Binary('x[2]'))) + (Binary('x[1]') \* Binary('x[2]'))))) + (Binary('x[0]') \*

(((1.000000 + (-1.000000 \* Binary('x[1]'))) + (-1.000000 \* Binary('x[2]'))) + (Binary('x[1]') \* Binary('x[2]')))))

Note: If worked out by hand this can be simplified greatly but the code above will still have the same affect

**Link every 7 variables to 1 slack variable. Then apply a penalty if the sum of the slack variables is not K**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Easier to understand and visualize | Requires adding the necessary slack variables to every variable in the program. Makes producing the model for the CC problem more difficult |
| The penalty for the slack variables is a lot simpler | Also links lots of variables |

Example of the slack variable penalty for a 3-variable system, deduced by me:

P(K- Binary('x[3]')- Binary('x[4]')- Binary('x[5]')) \*\* 2

**First Attempt**

I initially attempted to create code that implemented the first solution. The implementation itself ended up being successful however when the final constraint was used in the objective function it wasn’t quadratic and therefore couldn’t be converted into a QUBO.

After discussing how to approach this problem with Fujitsu Laboratories they advised that I implement the second solution as that is what they use. I am now going to attempt to implement the second solution while keeping the objective function quadratic.

**Second Attempt**

23/02/2022, 23:28 – Implementation of auxiliary variables was successful. Each group of variables is now linked to an auxiliary variable. The cardinality constraint on those variables is also implemented properly. The problem is that by linking the variables the objective function has stopped being quadratic and again won’t convert properly to a QUBO. In an attempt to fix this, I am going to refer to the original penalty in the main paper (Heuristics for cardinality constrained portfolio optimization, equation 19).

23/02/2022, 23:34 – The penalty the paper uses is an inequality penalty which already poses its own problems when trying to convert to a quadratic penalty. This is made trickier by the fact that 4 different variables are involved in the penalty and the inequality is based on integer values.