

# FLOW PLANNING

## ASSIGNMENT 2

COSC364-19S1 INTERNET TECHNOLOGY AND ENGINEERING

Will Cowper

ID: 81163265

Contribution: 50%

Jesse Sheehan

ID: 53366509

Contribution: 50%

May 30, 2019

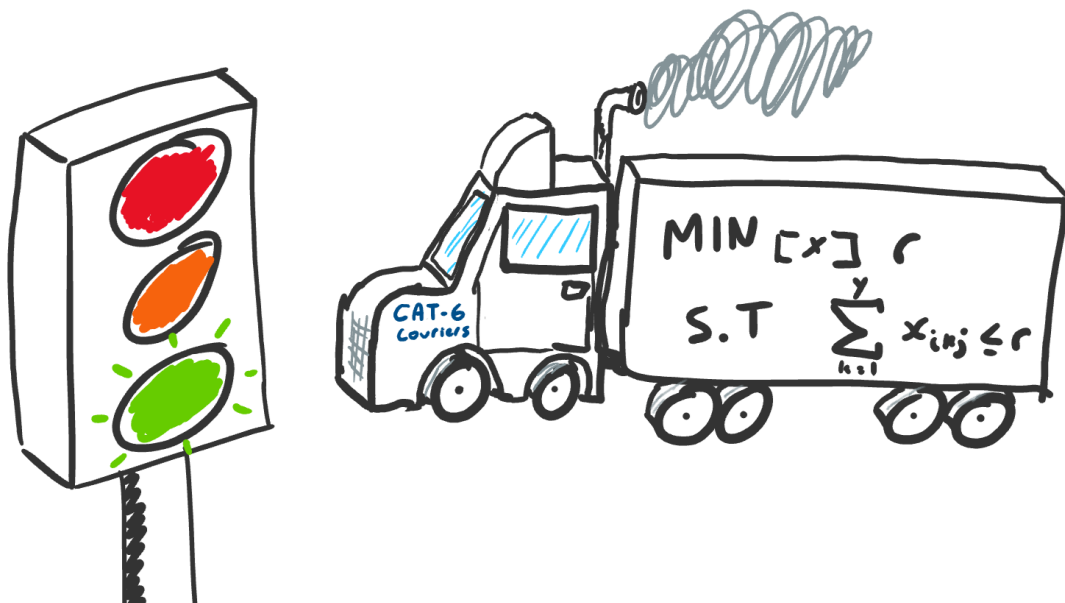


Figure 1: Traffic problems are not unique to computer networks.

# Plagiarism Declaration

This form needs to accompany your COSC 364 assignment submission.

I understand that plagiarism means taking someone else's work (text, program code, ideas, concepts) and presenting them as my own, without proper attribution. Taking someone else's work can include verbatim copying of text, figures/images, or program code, or it can refer to the extensive use of someone else's original ideas, algorithms or concepts.

I hereby declare that:

- My assignment is my own original work. I have not reproduced or modified code, figures/images, or writings of others without proper attribution. I have not used original ideas and concepts of others and presented them as my own.
- I have not allowed others to copy or modify my own code, figures/images, or writings. I have not allowed others to use original ideas and concepts of mine and present them as their own.
- I accept that plagiarism can lead to consequences, which can include partial or total loss of marks, no grade being awarded and other serious consequences, including notification of the University Proctor.

Name:

Will Cowper      Jesse Sheehan

Student ID:

81163265      53366500

Signature:

Date:

29-5-19      29/5/19

# 1 Problem Description

Given a network (figure 2) with  $X$  source nodes,  $Y$  transit nodes and  $Z$  destination nodes, a program was designed to generate an LP file that could be used by CPLEX to determine certain network characteristics.

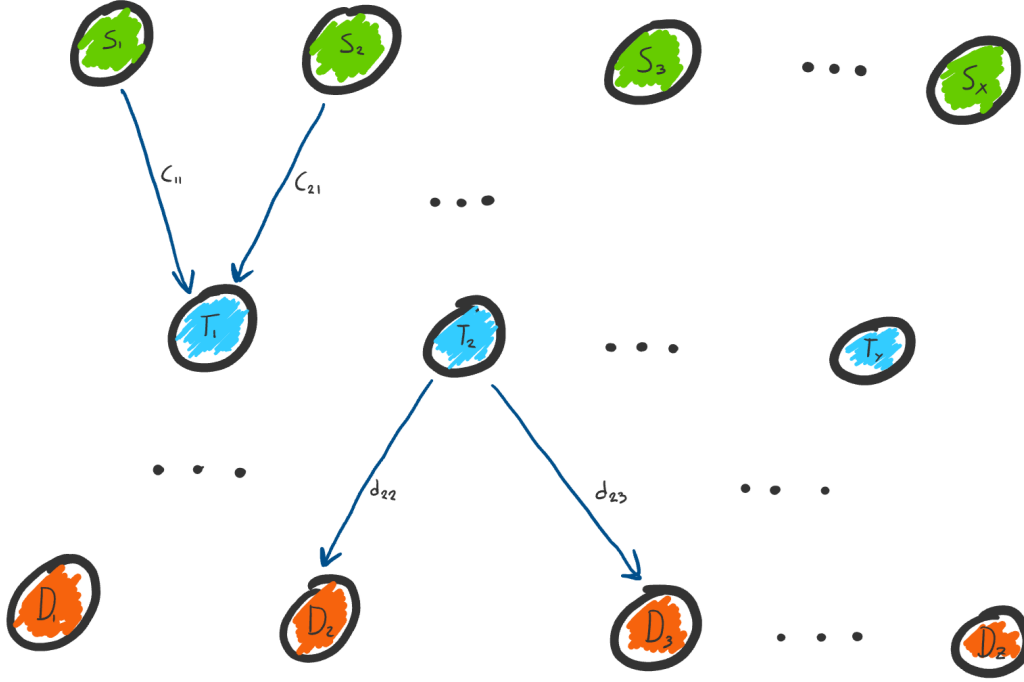


Figure 2: An example network showing nodes  $S_i$ ,  $T_k$  and  $D_j$  and links  $c_{ik}$  and  $d_{kj}$ .

Traffic travelling from  $S_i$  to  $D_j$  must travel through exactly 2 transit nodes with a total demand volume of  $h_{ij}$  (equation 10). Furthermore, the load upon each transit node must be balanced.

## 2 Problem Formulation

This problem was solved with the use of binary variable constraints (equations 6, 7 and 9) and the minimisation of our objective function (equation 1). All normal non-negativity constraints were applied (equations 11, 12, 13 and 14).

The following network properties were solved for:

- The capacities of each link (equations 3 and 4).
- The load on each transit node (equation 5).
- The value of each flow (equations 2 and 8).

**Notation:**

- $X$  is the number of source nodes.

- $Y$  is the number of transit nodes.
- $Z$  is the number of destination nodes.
- $S_i$  is the  $i$ th source node.
- $T_k$  is the  $k$ th transit node.
- $D_j$  is the  $j$ th destination node.
- $h_{ij}$  is the demand flow between  $S_i$  and  $D_j$ . This is equal to  $2i + j$ .
- $c_{ik}$  is the link capacity between  $S_i$  and  $T_k$ .
- $d_{kj}$  is the link capacity between  $T_k$  and  $D_j$ .
- $x_{ikj}$  is the decision variable associated with the path  $S_i$ - $T_k$ - $D_j$ .
- $u_{ikj}$  is the binary decision variable associated with  $x_{ikj}$ . These are required because  $h_{ij}$  must be split across exactly 2 transit nodes.
- $l_k$  is the load on  $T_k$ .

**Note:** Due to the limitations of the LP file format, many of the following equations must be rearranged for use in CPLEX. Most notably, there cannot be any variables on the right hand side of an equality or inequality.

## 2.1 Objective Function

$$\text{minimize}_{[x,c,d,r]} r \quad (1)$$

## 2.2 Constraints

$$\sum_{k=1}^Y x_{ikj} = h_{ij} \quad i \in \{1, \dots, X\}, j \in \{1, \dots, Z\} \quad (2)$$

$$\sum_{j=1}^Z x_{ikj} = c_{ik} \quad i \in \{1, \dots, X\}, k \in \{1, \dots, Y\} \quad (3)$$

$$\sum_{i=1}^X x_{ikj} = d_{kj} \quad k \in \{1, \dots, Y\}, j \in \{1, \dots, Z\} \quad (4)$$

$$\sum_{k=1}^Y x_{ikj} = l_k \quad i \in \{1, \dots, X\}, j \in \{1, \dots, Z\} \quad (5)$$

$$\sum_{k=1}^Y u_{ikj} = 2 \quad i \in \{1, \dots, X\}, j \in \{1, \dots, Z\} \quad (6)$$

$$x_{ikj} = \frac{u_{ikj} h_{ij}}{2} \quad i \in \{1, \dots, X\}, k \in \{1, \dots, Y\}, j \in \{1, \dots, Z\} \quad (7)$$

$$\sum_{i=1}^X \sum_{j=1}^Z x_{ikj} \leq r \quad k \in \{1, \dots, Y\} \quad (8)$$

$$u_{ikj} \in \{0, 1\} \quad i \in \{1, \dots, X\}, k \in \{1, \dots, Y\}, j \in \{1, \dots, Z\} \quad (9)$$

$$h_{ij} = 2i + j \quad i \in \{1, \dots, X\}, j \in \{1, \dots, Z\} \quad (10)$$

## 2.3 Non-Negativity Constraints

$$r \geq 0 \quad (11)$$

$$x_{ikj} \geq 0 \quad i \in \{1, \dots, X\}, k \in \{1, \dots, Y\}, j \in \{1, \dots, Z\} \quad (12)$$

$$c_{ik} \geq 0 \quad i \in \{1, \dots, X\}, k \in \{1, \dots, Y\} \quad (13)$$

$$d_{kj} \geq 0 \quad k \in \{1, \dots, Y\}, j \in \{1, \dots, Z\} \quad (14)$$

### 3 Results

LP files were generated with parameters  $X = Z = 9, Y \in \{3, 4, 5, 6, 7, 8\}$ . These were then processed with CPLEX, recording the time taken to solve each problem. Important data points were extracted from the CPLEX output and are listed in table 1.

Table 1: The raw data as extracted and processed from the CPLEX output.

Y	Time (ms)	Links	Load Spread	Max. $c_{ik}$	Max. $d_{kj}$
3	81.6132	52	0.0	94.500000	73.500000
4	117.3323	68	0.5	8.500000	9.500000
5	285.8704	83	0.0	9.500000	9.500000
6	443.1351	101	0.0	83.000000	9.500000
7	223.422	118	1.5	9.500000	9.500000
8	987.7775	134	0.5	9.500000	9.500000

An analysis of these results confirms many assumptions that were made about the problem. The number of non-zero link capacities increases linearly (figure 4), the transit node load spread is very close to (if not exactly) zero for most networks (figure 5), and the amount of time taken to solve the problem increases (almost) non-linearly as the number of transit nodes increases (figure 3). It is important to note that the load on the transit nodes was only able to be balanced in three of the networks. **Give a reason why.**

The most obvious feature of the results is the data for the  $Y = 7$  network. It appears to be an outlier as it takes less time to solve than the  $Y = 5$  network and has the highest transit node load spread by a factor of 3. **Explain this behaviour!!!**

The second odd feature of the data is the highest link capacities of the  $Y = 3$  and  $Y = 6$  networks (figure 6). **Also explain this behaviour!!!**

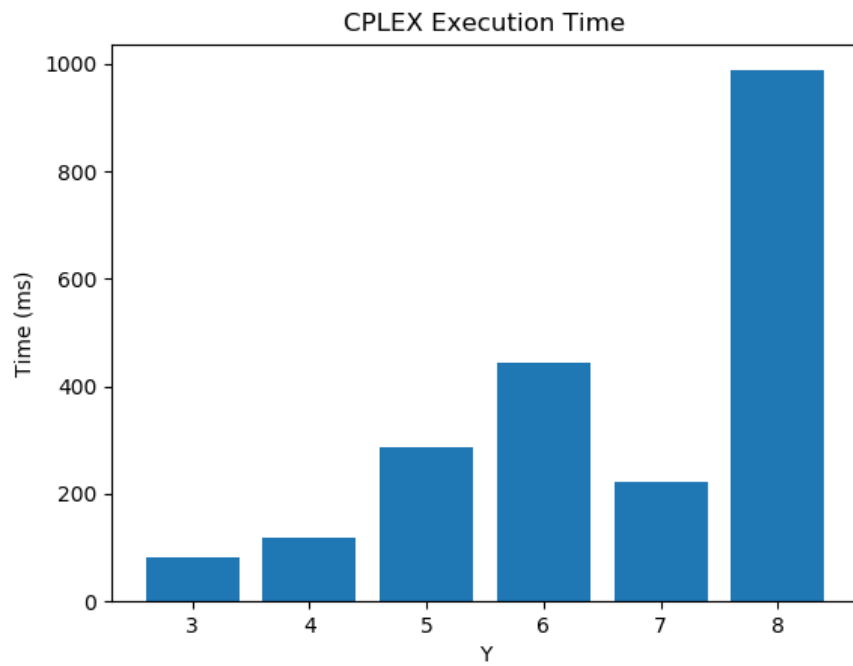


Figure 3: The time taken to execute the LP file in CPLEX for each network.

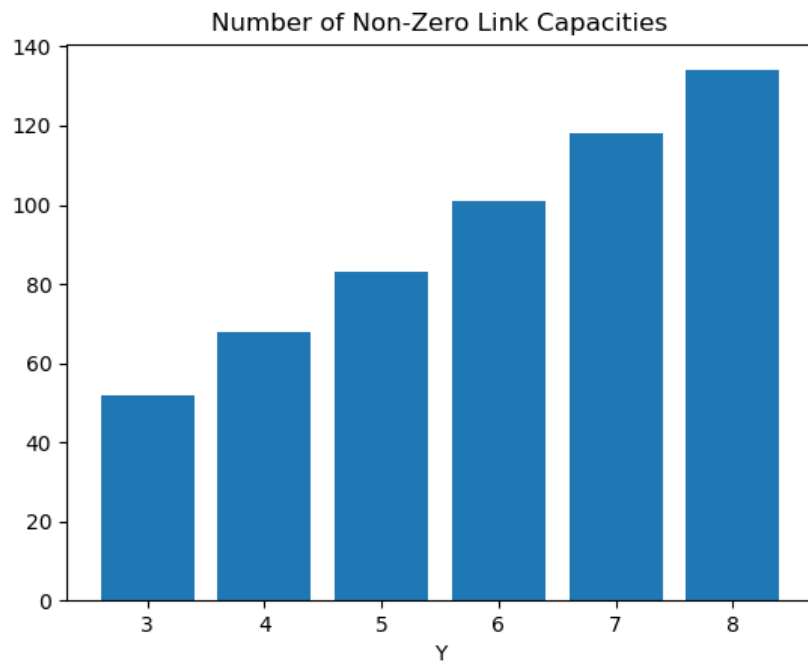


Figure 4: The number of non-zero link capacities in each network.

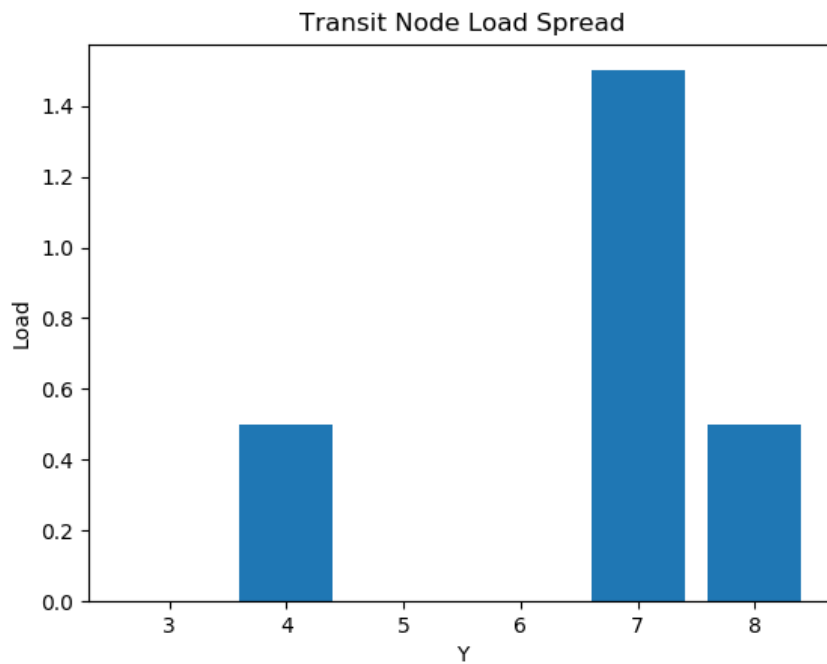


Figure 5: The amount of spread in the load for all transit nodes in each network.

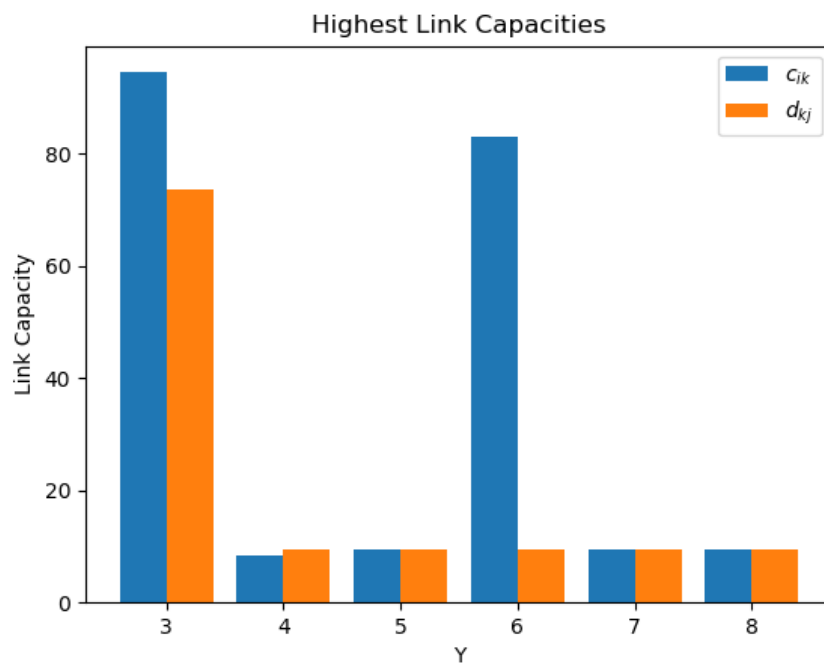


Figure 6: The highest capacity links for each network. Both the  $C_{ik}$  and  $D_{kj}$  links are listed.



## 4 Appendix

### 4.1 Source Code

#### 4.1.1 src/lp\_gen.py

This script is responsible for producing a valid LP file from the given command line parameters.

```

import inspect
import functools
import sys
import os.path

__TITLE__ = "COSC-364 Assignment 2 LP Generator"
__AUTHORS__ = [("Will Cowper", "81163265"), ("Jesse Sheehan", "53366509")]

# Change these variables to alter the behaviour of the LP file generator
PATH_SPLIT = 2

def DEMANDFLOW(i, j): return 2 * i + j

TEMPLATE = """\
\\ {}, LP Output File
\\ Written by {}
\\ Parameters: X={}, Y={}, Z={}, Split={}, Demand={}

MINIMIZE
\t r

SUBJECT TO

\t \t DEMAND CONSTRAINTS
\t {}

\t \t CAPACITY CONSTRAINTS FOR LINKS BETWEEN SOURCE AND TRANSIT NODES
\t {}

\t \t CAPACITY CONSTRAINTS FOR LINKS BETWEEN TRANSIT AND DESTINATION NODES
\t {}

\t \t OBJECTIVE FUNCTION LOAD CONSTRAINTS
\t {}

\t \t TRANSIT NODE LOAD CONSTRAINTS
\t {}

\t \t BINARY VARIABLE AND DECISION VARIABLE CONSTRAINTS
\t {}

\t \t BINARY VARIABLE CONSTRAINTS (ONLY 2 ACTIVE TRANSIT NODES)
\t {}

BOUNDS

\t \t NON-NEGATIVITY CONSTRAINTS

```

```

50 \tr >= 0
51 \t{}
52
53 BIN
54
55 \t\\ BINARY VARIABLES
56 \t{}
57
58 END
59 """
60
61 # DEFINE SOME UTILITY FUNCTIONS
62
63
64 def get_lp_filename(x, y, z):
65     """ Returns the filename that the LP data should be saved to. """
66     return "problem_{0}-{1}-{2}.lp".format(x, y, z)
67
68
69 def crange(first, last):
70     """ Returns a list of characters between the two characters passed in (
71     inclusive).
72     >>> crange('A', 'C')
73     ['A', 'B', 'C']
74     >>> crange('A', 'A')
75     ['A']
76     """
77     if ord(first) > ord(last):
78         raise ValueError("last must come after first")
79
80     else:
81         return [chr(i) for i in range(ord(first), ord(last) + 1)]
82
83
84 def repeat(obj, n):
85     """ Returns a list with obj repeated n times.
86     >>> repeat(1, 1)
87     [1]
88     >>> repeat(42, 0)
89     []
90     >>> repeat(5, 4)
91     [5, 5, 5, 5]
92     >>> repeat([1, 2], 2)
93     [[1, 2], [1, 2]]
94     """
95     return [obj for _ in range(n)]
96
97
98 def perms(lists):
99     """ Returns all the permutations of the elements.
100     >>> perms([])
101     []
102     >>> perms(['a', 'b', 'c'])
103     [('a',), ('b',), ('c',)]
104     >>> perms(['a', 'b', 'c'], ['x', 'y', 'z'])
105     [('a', 'x'), ('a', 'y'), ('a', 'z'), ('b', 'x'), ('b', 'y'), ('b', 'z'),
106     ('c', 'x'), ('c', 'y'), ('c', 'z')]
107     """

```

```

106     if len(lists) == 0:
107         return []
108
109     elif (len(lists) == 1):
110         return [(x,) for x in lists[0]]
111
112     else:
113         return [(x,) + y for x in lists[0] for y in perms(lists[1:])]
114
115 def concat(permutations):
116     """ Returns the permutations concatenated as strings.
117     """
118     >>> concat(perms([[ 'a', 'b', 'c ']]))
119     [ 'a', 'b', 'c ' ]
120     >>> concat(perms([[ 'a', 'b', 'c '], [ 'x', 'y', 'z ']]))
121     [ 'ax', 'ay', 'az', 'bx', 'by', 'bz', 'cx', 'cy', 'cz ' ]
122     """
123     return [functools.reduce(lambda x, y: x + str(y), p, '') for p in
124             permutations]
125
126 def get_function_source(fn):
127     src = inspect.getsource(fn)
128     return src[str(src).index('return')+7:]
129
130 def get_lines(strings):
131     return '\n\t'.join(strings)
132
133 # DEFINE SOME FUNCTIONS SPECIFIC TO THE PROBLEM
134
135 def get_nodes(x, y, z):
136     """ Returns a tuple containing the source, transit and destination node
137     ids as integers. """
138     s = list(range(1, x + 1))
139     t = list(range(1, y + 1))
140     d = list(range(1, z + 1))
141     return s, t, d
142
143 def get_demand_constraints(s, t, d):
144     """ Returns a list of demand constraints. """
145     return [' + '.join(["x-{}{}{}".format(i, k, j) for k in t]) + ' =
146             {}'.format(DEMANDFLOW(i, j))
147             for (i, j) in perms([s, d])]
148
149 def get_source_transit_capacity_constraints(s, t, d):
150     """ Returns a list of capacity constraints for the links between the
151     source and transit nodes. """
152     return \
153         [' + '.join(["x-{}{}{}".format(i, k, j) for j in d]) +
154          ' - c-{}{} = 0'.format(i, k) for (i, k) in perms([s, t])]
155
156 def get_transit_destination_capacity_constraints(s, t, d):
157     """ Returns a list of capacity constraints for the links between the

```

```

transit and destination nodes. """
160     return \
        [' + '.join(["x_{0}{1}{2}".format(i, k, j) for i in s]) +
162         ' - d_{0}{1} = 0'.format(k, j) for (k, j) in perms([t, d])]

164 def get_transit_load_constraints(s, t, d):
166     """ Returns the list of transit load constraints. """
168     return [' + '.join(["x_{0}{1}{2}".format(i, k, j) for (i, j) in perms([
s, d])]) +
170         ' - l_{0} = 0'.format(k) for k in t]

172 def get_objective_function_load_constraints(s, t, d):
174     """ Returns the list of objective function load constraints. """
176     return [' + '.join(["c_{0}{1}".format(i, j) for i in s]) +
178         ' - r <= 0' for j in d]

180 def get_binary_and_decision_variable_constraints(s, t, d):
182     """ Returns the binary and decision variable constraints. """
184     return ['{3} x_{0}{1}{2} - {4} u_{0}{1}{2} = 0'.format(i, k, j,
PATH_SPLIT, DEMAND_FLOW(i, j)) for (i, k, j) in perms([s, t, d])]

186 def get_binary_constraints(s, t, d):
188     """ Returns a list of binary variable constraints. """
190     return [' + '.join(["u_{0}{1}{2}".format(i, k, j) for k in t]) + ' = {'
'.format(PATH_SPLIT)
192         for (i, j) in perms([s, d])]

194 def get_binary_variables(s, t, d):
196     """ Returns a list of binary variables. """
198     return ["u_{0}{1}{2}".format(i, k, j) for (i, k, j) in perms([s, t, d])
]

200 def get_non_negativity_constraints(s, t, d):
202     """ Returns a list of non-negativity constraints. """
204     return ["x_{0}{1}{2} >= 0".format(i, k, j) for (i, k, j) in perms([s, t,
d])] + ["c_{0}{1} >= 0".format(i, k) for (i, k) in perms([s, t])] + ["
d_{0}{1} >= 0".format(k, j) for (k, j) in perms([t, d])]

206 def generate_lp_file(title, authors, x, y, z):
208     """ Returns the LP file contents as per the project specification. """
210     s, t, d = get_nodes(x, y, z)

    demand_constraints = get_lines(get_demand_constraints(s, t, d))
    source_transit_capacity_constraints = get_lines(
        get_source_transit_capacity_constraints(s, t, d))
    transit_destination_capacity_constraints = get_lines(
        get_transit_destination_capacity_constraints(s, t, d))
    non_negativity_constraints = get_lines(get_non_negativity_constraints(
        s, t, d))
    objective_function_load_constraints = get_lines(
        get_objective_function_load_constraints(s, t, d))

```

```

transit_load_constraints = get_lines(
    get_transit_load_constraints(s, t, d))
binary_and_decision_constraints = get_lines(
    get_binary_and_decision_variable_constraints(s, t, d))
binary_variable_constraints = get_lines(get_binary_constraints(s, t, d)
)
binary_variables = get_lines(get_binary_variables(s, t, d))

return TEMPLATE.format(
    title ,
    authors ,
    x,
    y,
    z,
    PATH_SPLIT,
    get_function_source(DEMANDFLOW) ,
    demand_constraints ,
    source_transit_capacity_constraints ,
    transit_destination_capacity_constraints ,
    objective_function_load_constraints ,
    transit_load_constraints ,
    binary_and_decision_constraints ,
    binary_variable_constraints ,
    non_negativity_constraints ,
    binary_variables)

# DEFINE SOME HELPERS FOR GETTING THE THING RUNNING

def print_version():
    print( '{0} by {1}'.format(__TITLE__, get_author_string()))

def print_usage():
    print( 'Usage: {0} <x> <y> <z> [output directory]'.format(sys.argv[0]))

def get_problem_parameters():
    """ Returns a tuple containing the x, y and z parameters. """
    try:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
        z = int(sys.argv[3])
    except:
        print_usage()
        exit(-1)

    if x <= 0:
        print("Error: x must be strictly positive")
        exit(-1)

    if x >= 10:
        print("Error: x must be less than ten")
        exit(-1)

    if y <= 0:
        print("Error: y must be strictly positive")
        exit(-1)

```

```

268     if y >= 10:
270         print("Error: y must be less than ten")
272         exit(-1)
274     if z <= 0:
276         print("Error: z must be strictly positive")
278         exit(-1)
280     if z >= 10:
282         print("Error: z must be less than ten")
284         exit(-1)
286     return x, y, z
288
290 def save_lp_file(filename, data):
292     try:
294         f = open(filename, 'w')
296         f.write(data)
298         f.close()
300     except:
302         print("Error: could not save file '{0}'".format(filename))
304         exit(-1)
306
308 def get_author_string():
310     return ', '.join(
312         ["{0} ({1})".format(name, sid) for (name, sid) in _AUTHORS_])
314
316 def main():
318     print_version()
320     if len(sys.argv) != 4 and len(sys.argv) != 5:
322         print_usage()
324         exit(-1)
326     else:
328         output_dir = '.'
330         if len(sys.argv) == 5:
332             output_dir = sys.argv[4]
334
336         x, y, z = get_problem_parameters()
338         data = generate_lp_file(_TITLE_, get_author_string(), x, y, z)
340         filename = os.path.join(output_dir, get_lp_filename(x, y, z))
342         save_lp_file(filename, data)
344         print("Success: saved as '{0}'".format(filename))
346
348 if __name__ == "__main__":
350     main()

```

../src/lp-gen.py

#### 4.1.2 src/lp\_csv.py

This script is responsible for converting the output of the CPLEX log files into a single CSV file for further processing.

```

import csv
2 import sys
import os.path
4

6 def csvWrite(data):
    with open(sys.argv[2], 'a', newline='') as csvFile:
8         writer = csv.writer(csvFile)
            writer.writerow(data)
10

12 def floatmap(enumerable):
    return list(map(lambda x: float(x), enumerable))
14

16 def openFile(Y):
    with open(os.path.join(sys.argv[1], '{0}.txt'.format(Y)), 'r') as
in_file:
18         stripped = [line.strip() for line in in_file.readlines()]
            lines = [line for line in stripped if line]
20             data = []
                # Y
22                 data.append(Y)
                    # elapsed time
24                     data.append(max(parseFile("elapsed_", lines)))
                        # no of non-zero c and d links
26                         data.append(len(parseFile("c_", lines)) + len(parseFile("d_", lines
)))
                            # transit load spread (largest_transit_node_load -
smallest_transit_node_load)
28                             data.append(max(floatmap(parseFile("l_", lines))) -
min(floatmap(parseFile("l_", lines))))
30                                 # highest cap c network
                                    data.append(max(parseFile("c_", lines)))
42                                     # highest cap d network
                                        data.append(max(parseFile("d_", lines)))
64                                         csvWrite(data)
66
68 '''Returns a list of all values that start with the given string'''
70
72 def parseFile(string, lines):
    values = []
42     for line in lines:
44         if line.startswith(string):
            values.append(line.split()[1])
46
48     return values
50

52 if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Usage: {0} <input directory> <csv file>".format(sys.argv[0])
        )
        exit(-1)

```

```
54 # delete the CSV, otherwise we will append to it
    os.unlink(sys.argv[2])

56
    openFile(3)
58 openFile(4)
    openFile(5)
60 openFile(6)
    openFile(7)
62 openFile(8)

64 print("Saved CSV data to '{}'".format(sys.argv[2]))
```

../src/lp\_csv.py

### 4.1.3 src/lp\_graph.py

This script is responsible for reading the CSV file and producing several graphs.

```
import csv
2 import sys
import os.path
4 import numpy as np

6 try:
    import matplotlib.pyplot as plt
8 except:
    print("Error: could not load 'matplotlib'. Install with 'pip install
    matplotlib' and then try again.")
10 exit(-1)

12
def get_data(data, key):
14     return list(map(lambda d: d[key], data))

16
def get_time(data):
18     return get_data(data, "time")

20
def get_len_nonzero_links(data):
22     return get_data(data, "len_links")

24
def get_transit_load_spread(data):
26     return get_data(data, "load_spread")

28
def get_max_cap_c(data):
30     return get_data(data, "max_cap_c")

32
def get_max_cap_d(data):
34     return get_data(data, "max_cap_d")

36
def get_Y(data):
38     return get_data(data, "Y")
```



```

40
41 def save_execution_time_plot(filename, data):
42     """ Saves a plot of execution time. """
43     plt.bar(get_Y(data), get_time(data))
44     plt.xlabel("Y")
45     plt.ylabel("Time (ms)")
46     plt.title("CPLEX Execution Time")
47     plt.savefig(filename)
48     plt.close()
49     print("Saved '{}'".format(filename))
50
51
52 def save_num_nonzero_links_plot(filename, data):
53     """ Saves a plot of the number of non-zero links. """
54     plt.bar(get_Y(data), get_len_nonzero_links(data))
55     plt.xlabel("Y")
56     plt.ylabel("")
57     plt.title("Number of Non-Zero Link Capacities")
58     plt.savefig(filename)
59     plt.close()
60     print("Saved '{}'".format(filename))
61
62
63 def save_transit_load_spread_plot(filename, data):
64     """ Saves a plot of the transit load spread. """
65     plt.bar(get_Y(data), get_transit_load_spread(data))
66     plt.xlabel("Y")
67     plt.ylabel("Load")
68     plt.title("Transit Node Load Spread")
69     plt.savefig(filename)
70     plt.close()
71     print("Saved '{}'".format(filename))
72
73
74 def save_highest_capacity_links_plot(filename, data):
75     """ Saves a plot of the transit load spread. """
76     width = 0.4
77     Ys = np.array(get_Y(data))
78     cs = plt.bar(Ys, get_max_cap_c(data), width, label="$c_{ik}$")
79     ds = plt.bar(Ys + width, get_max_cap_d(data), width, label="$d_{kj}$")
80     plt.xticks(Ys + width / 2, map(lambda x: int(x), Ys))
81     plt.legend(handles=[cs, ds])
82     plt.xlabel("Y")
83     plt.ylabel("Link Capacity")
84     plt.title("Highest Link Capacities")
85     plt.savefig(filename)
86     plt.close()
87     print("Saved '{}'".format(filename))
88
89
90 def get_data_from_csv(csv_filename):
91     """ Returns an array of dictionaries containing the CSV data. """
92     with open(csv_filename, newline='') as csv_file:
93         csv_reader = csv.DictReader(csv_file, fieldnames=[
94             "Y", "time", "len-links", "load-spread",
95             "max-cap-c", "max-cap-d"])
96         rows = []
97         for row in csv_reader:

```

```

100         d = {}
101         for key in row:
102             d[key] = float(row[key])
103         rows.append(d)
104     return rows
105
106 def convert_csv_to_images(csv_filename, output_folder):
107     """ Converts the data from the CSV into a set of graphs. """
108     data = get_data_from_csv(csv_filename)
109     base_filename = os.path.splitext(os.path.join(
110         output_folder, os.path.basename(csv_filename)))[0]
111
112     save_execution_time_plot(base_filename + "_time.png", data)
113     save_num_nonzero_links_plot(base_filename + "_num_nonzero_links.png",
114                                data)
115     save_transit_load_spread_plot(
116         base_filename + "_transit_load_spread.png", data)
117     save_highest_capacity_links_plot(
118         base_filename + "_highest_capacity_links.png", data)
119
120 def print_usage():
121     print("Usage: {0} <csv file> <output folder>")
122
123 if __name__ == "__main__":
124     if len(sys.argv) != 3:
125         print_usage()
126         exit(-1)
127
128     convert_csv_to_images(sys.argv[1], sys.argv[2])

```

../src/lp-graph.py

#### 4.1.4 output.sh

This BASH script is responsible for executing the other scripts as well as timing and running CPLEX (under the Linux operating system).

```

1  #!/bin/bash
2  for y in 3 4 5 6 7 8
3  do
4      python3 src/lp-gen.py 9 $y 9 lp-files
5      start=$(date +%s%N)
6      cplex -c "read lp-files/problem_9-${y}_9.lp" "optimize" "display
7      solution variables -" > cplex_logs/$y.txt
8      end=$(date +%s%N)
9      duration=$(expr $end - $start)
10     duration=$(expr $duration / 1000000)
11     echo -e "\nelapsed_time: $duration ms" >> cplex_logs/$y.txt
12 done
13
14 python3 src/lp-csv.py cplex_logs cplex_data.csv
15 python3 src/lp-graph.py cplex_data.csv graphs

```

../output.sh

### 4.1.5 output.ps1

This PowerShell script is responsible for executing the other scripts as well as timing and running CPLEX (under the Windows operating system).

```

2 For ($i=3; $i -le 8; $i++) {
    python src/lp_gen.py 9 $i 9 lp_files
    $perf = Measure-Command -Expression {$data = cplex -c ("read lp_files/
4 problem_9_" + $i + "_9.lp") "optimize" "display solution variables -"}
    $ms = $perf.TotalMilliseconds
    [System.IO.File]::WriteAllLines("cplex_logs/$i.txt", $data + "
6     nelapsed_time: $ms ms")
}

8 python src/lp_csv.py cplex_logs lp_files/cplex_data.csv
python src/lp_graph.py lp_files/cplex_data.csv graphs

```

../output.ps1

## 4.2 Generated LP File

### 4.2.1 lp\_files/problem.3.2.4.lp

```

2 \ COSC-364 Assignment 2 LP Generator, LP Output File
\ Written by Will Cowper (81163265), Jesse Sheehan (53366509)
\ Parameters: X=3, Y=2, Z=4, Split=2, Demand=2 * i + j
4
6 MINIMIZE
    r
8
10 SUBJECT TO
    \ DEMAND CONSTRAINTS
12 x_111 + x_121 = 3
    x_112 + x_122 = 4
14 x_113 + x_123 = 5
    x_114 + x_124 = 6
16 x_211 + x_221 = 5
    x_212 + x_222 = 6
18 x_213 + x_223 = 7
    x_214 + x_224 = 8
20 x_311 + x_321 = 7
    x_312 + x_322 = 8
22 x_313 + x_323 = 9
    x_314 + x_324 = 10
24
    \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN SOURCE AND TRANSIT NODES
26 x_111 + x_112 + x_113 + x_114 - c_11 = 0
    x_121 + x_122 + x_123 + x_124 - c_12 = 0
28 x_211 + x_212 + x_213 + x_214 - c_21 = 0
    x_221 + x_222 + x_223 + x_224 - c_22 = 0
30 x_311 + x_312 + x_313 + x_314 - c_31 = 0
    x_321 + x_322 + x_323 + x_324 - c_32 = 0
32
    \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN TRANSIT AND DESTINATION NODES
34 x_111 + x_211 + x_311 - d_11 = 0

```

```

36  x_112 + x_212 + x_312 - d_12 = 0
    x_113 + x_213 + x_313 - d_13 = 0
    x_114 + x_214 + x_314 - d_14 = 0
38  x_121 + x_221 + x_321 - d_21 = 0
    x_122 + x_222 + x_322 - d_22 = 0
40  x_123 + x_223 + x_323 - d_23 = 0
    x_124 + x_224 + x_324 - d_24 = 0
42
    \ OBJECTIVE FUNCTION LOAD CONSTRAINTS
44  c_11 + c_21 + c_31 - r <= 0
    c_12 + c_22 + c_32 - r <= 0
46  c_13 + c_23 + c_33 - r <= 0
    c_14 + c_24 + c_34 - r <= 0
48
    \ TRANSIT NODE LOAD CONSTRAINTS
50  x_111 + x_112 + x_113 + x_114 + x_211 + x_212 + x_213 + x_214 + x_311 +
    x_312 + x_313 + x_314 - l_1 = 0
    x_121 + x_122 + x_123 + x_124 + x_221 + x_222 + x_223 + x_224 + x_321 +
    x_322 + x_323 + x_324 - l_2 = 0
52
    \ BINARY VARIABLE AND DECISION VARIABLE CONSTRAINTS
54  2 x_111 - 3 u_111 = 0
    2 x_112 - 4 u_112 = 0
56  2 x_113 - 5 u_113 = 0
    2 x_114 - 6 u_114 = 0
58  2 x_121 - 3 u_121 = 0
    2 x_122 - 4 u_122 = 0
60  2 x_123 - 5 u_123 = 0
    2 x_124 - 6 u_124 = 0
62  2 x_211 - 5 u_211 = 0
    2 x_212 - 6 u_212 = 0
64  2 x_213 - 7 u_213 = 0
    2 x_214 - 8 u_214 = 0
66  2 x_221 - 5 u_221 = 0
    2 x_222 - 6 u_222 = 0
68  2 x_223 - 7 u_223 = 0
    2 x_224 - 8 u_224 = 0
70  2 x_311 - 7 u_311 = 0
    2 x_312 - 8 u_312 = 0
72  2 x_313 - 9 u_313 = 0
    2 x_314 - 10 u_314 = 0
74  2 x_321 - 7 u_321 = 0
    2 x_322 - 8 u_322 = 0
76  2 x_323 - 9 u_323 = 0
    2 x_324 - 10 u_324 = 0
78
    \ BINARY VARIABLE CONSTRAINTS (ONLY 2 ACTIVE TRANSIT NODES)
80  u_111 + u_121 = 2
    u_112 + u_122 = 2
82  u_113 + u_123 = 2
    u_114 + u_124 = 2
84  u_211 + u_221 = 2
    u_212 + u_222 = 2
86  u_213 + u_223 = 2
    u_214 + u_224 = 2
88  u_311 + u_321 = 2
    u_312 + u_322 = 2
90  u_313 + u_323 = 2

```

$$u_{314} + u_{324} = 2$$

## BOUNDS

\ NON-NEGATIVITY CONSTRAINTS

$$r \geq 0$$

$$x_{111} \geq 0$$

$$x_{112} \geq 0$$

$$x_{113} \geq 0$$

$$x_{114} \geq 0$$

$$x_{121} \geq 0$$

$$x_{122} \geq 0$$

$$x_{123} \geq 0$$

$$x_{124} \geq 0$$

$$x_{211} \geq 0$$

$$x_{212} \geq 0$$

$$x_{213} \geq 0$$

$$x_{214} \geq 0$$

$$x_{221} \geq 0$$

$$x_{222} \geq 0$$

$$x_{223} \geq 0$$

$$x_{224} \geq 0$$

$$x_{311} \geq 0$$

$$x_{312} \geq 0$$

$$x_{313} \geq 0$$

$$x_{314} \geq 0$$

$$x_{321} \geq 0$$

$$x_{322} \geq 0$$

$$x_{323} \geq 0$$

$$x_{324} \geq 0$$

$$c_{11} \geq 0$$

$$c_{12} \geq 0$$

$$c_{21} \geq 0$$

$$c_{22} \geq 0$$

$$c_{31} \geq 0$$

$$c_{32} \geq 0$$

$$d_{11} \geq 0$$

$$d_{12} \geq 0$$

$$d_{13} \geq 0$$

$$d_{14} \geq 0$$

$$d_{21} \geq 0$$

$$d_{22} \geq 0$$

$$d_{23} \geq 0$$

$$d_{24} \geq 0$$

## BIN

\ BINARY VARIABLES

$u_{111}$

$u_{112}$

$u_{113}$

$u_{114}$

$u_{121}$

$u_{122}$

$u_{123}$

$u_{124}$

$u_{211}$

$u_{212}$

```
150  u_213
      u_214
152  u_221
      u_222
      u_223
154  u_224
      u_311
156  u_312
      u_313
158  u_314
      u_321
160  u_322
      u_323
162  u_324
164  END
```

../lp\_files/problem\_3\_2\_4.lp