

TRAFFIC PLANNING

ASSIGNMENT 2

COSC364-19S1 INTERNET TECHNOLOGY AND ENGINEERING

Will Cowper

ID: 81163265

Contribution: 50%

Jesse Sheehan

ID: 53366509

Contribution: 50%

May 29, 2019

1 Problem Formulation

Given a network (figure 1) 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.

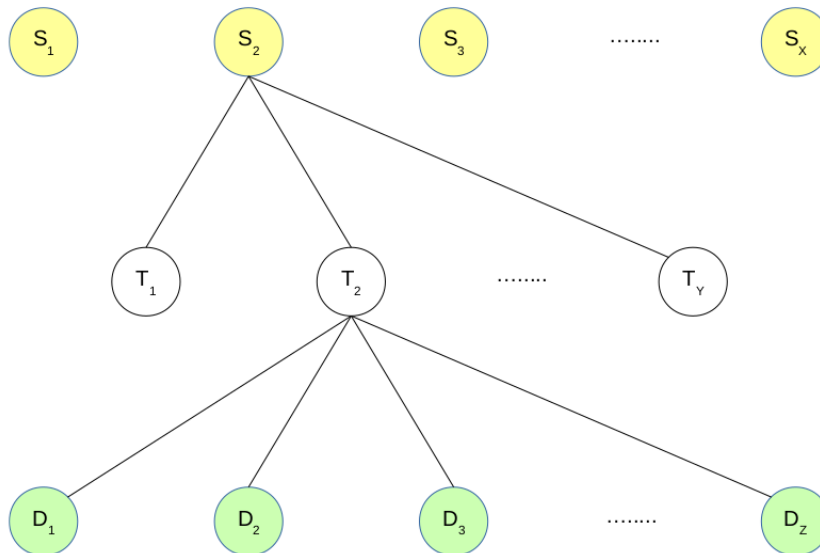


Figure 1: An example network (A. Willig, April 2019).

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. 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.

1.1 Objective Function

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

1.2 Constraints

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

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

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

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

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

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

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

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

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

1.3 Non-Negativity Constraints

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

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

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

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

2 Results

3 Appendix

3.1 Source Code

3.1.1 src/___main___py

```
import sys
2
from lp_gen import generate_lp_file
4 from lp_utils import get_lp_filename, run_cplex

6 __TITLE__ = "COSC-364 Assignment 2"
__AUTHORS__ = [("Will Cowper", "81163265"), ("Jesse Sheehan", "53366509")]
8

10 def print_version():
    print(' {0} by {1}'.format(__TITLE__, get_author_string()))
12

14 def print_usage():
    print('Usage: {0} <x> <y> <z>'.format(sys.argv[0]))
16

18 def get_problem_parameters():
    """ Returns a tuple containing the x, y and z parameters. """
20     try:
        x = int(sys.argv[1])
22         y = int(sys.argv[2])
        z = int(sys.argv[3])
24     except:
        print_usage()
        exit(-1)
26

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

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

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

40     return x, y, z

42
def save_lp_file(filename, data):
44     try:
        f = open(filename, 'w')
46         f.write(data)
        f.close()
48     except:
        print("Error: could not save file '{0}'".format(filename))
```

```

50         exit(-1)

52 def get_author_string():
53     return ', '.join(
54         ["{0} ({1})".format(name, sid) for (name, sid) in _AUTHORS_])

56 def main():
57     print_version()
58     if len(sys.argv) != 4:
59         print_usage()
60         exit(-1)
61     else:
62         x, y, z = get_problem_parameters()
63         data = generate_lp_file(_TITLE_, get_author_string(), x, y, z)
64         filename = get_lp_filename(x, y, z)
65         save_lp_file(filename, data)
66         print("Success: saved as '{0}'".format(filename))
67         run_cplex(filename)
68
70 if __name__ == "__main__":
71     main()

```

../src/_main_.py

3.1.2 src/lp_utils.py

```

import functools
2 import subprocess
import inspect
4
6 def get_lp_filename(x, y, z):
7     """ Returns the filename that the LP data should be saved to. """
8     return "problem-{0}-{1}-{2}.lp".format(x, y, z)
10
12 def run_cplex(filename):
13     """ Runs cplex on the LP file. """
14     subprocess.run(
15         ['cplex', '-c', "read {0}".format(filename), "optimize", "display solution variables -"])
16
18 def crange(first, last):
19     """ Returns a list of characters between the two characters passed in (
20     inclusive).
21     >>> crange('A', 'C')
22     ['A', 'B', 'C']
23     >>> crange('A', 'A')
24     ['A']
25     """
26     if ord(first) > ord(last):
27         raise ValueError("last must come after first")
28     else:
29         return [chr(i) for i in range(ord(first), ord(last) + 1)]

```

```

30 def repeat(obj, n):
31     """ Returns a list with obj repeated n times.
32     >>> repeat(1, 1)
33     [1]
34     >>> repeat(42, 0)
35     []
36     >>> repeat(5, 4)
37     [5, 5, 5, 5]
38     >>> repeat([1, 2], 2)
39     [[1, 2], [1, 2]]
40     """
41     return [obj for _ in range(n)]
42
43
44 def perms(lists):
45     """ Returns all the permutations of the elements.
46     >>> perms([])
47     []
48     >>> perms(['a', 'b', 'c'])
49     [('a',), ('b',), ('c',)]
50     >>> perms(['a', 'b', 'c'], ['x', 'y', 'z'])
51     [('a', 'x'), ('a', 'y'), ('a', 'z'), ('b', 'x'), ('b', 'y'), ('b', 'z'),
52     ('c', 'x'), ('c', 'y'), ('c', 'z')]
53     """
54     if len(lists) == 0:
55         return []
56
57     elif len(lists) == 1:
58         return [(x,) for x in lists[0]]
59
60     else:
61         return [(x,) + y for x in lists[0] for y in perms(lists[1:])]
62
63
64 def concat(permutations):
65     """ Returns the permutations concatenated as strings.
66     >>> concat(perms(['a', 'b', 'c']))
67     ['a', 'b', 'c']
68     >>> concat(perms(['a', 'b', 'c'], ['x', 'y', 'z']))
69     ['ax', 'ay', 'az', 'bx', 'by', 'bz', 'cx', 'cy', 'cz']
70     """
71     return [functools.reduce(lambda x, y: x + str(y), p, '') for p in
72             permutations]
73
74 def get_function_source(fn):
75     src = inspect.getsource(fn)
76     return src[src.index(':')+2:]
77
78 def get_lines(strings):
79     return '\n\t'.join(strings)
80
81 if __name__ == "__main__":
82     import doctest
83     doctest.testmod()

```

../src/lp_utils.py

3.1.3 src/lp_gen.py

```

1 from lp_utils import perms, concat, get_lines, get_function_source
2
3 # Change these variables to alter the behaviour of the LP file generator
4 PATH_SPLIT = 2
5 DEMAND_FLOW = lambda i, j: 2 * i + j
6
7 TEMPLATE = """\
8 \ \ {}, LP Output File
9 \ \ Written by {}
10 \ \ Parameters: X={}, Y={}, Z={}, Split={}, Demand={}
11
12 MINIMIZE
13 \ tr
14
15 SUBJECT TO
16
17 \ t \ \ DEMAND CONSTRAINTS
18 \ t {}
19
20 \ t \ \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN SOURCE AND TRANSIT NODES
21 \ t {}
22
23 \ t \ \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN TRANSIT AND DESTINATION NODES
24 \ t {}
25
26 \ t \ \ OBJECTIVE FUNCTION LOAD CONSTRAINTS
27 \ t {}
28
29 \ t \ \ TRANSIT NODE LOAD CONSTRAINTS
30 \ t {}
31
32 \ t \ \ BINARY VARIABLE AND DECISION VARIABLE CONSTRAINTS
33 \ t {}
34
35 \ t \ \ BINARY VARIABLE CONSTRAINTS (ONLY 2 ACTIVE TRANSIT NODES)
36 \ t {}
37
38 BOUNDS
39
40 \ t \ \ NON-NEGATIVITY CONSTRAINTS
41 \ tr >= 0
42 \ t {}
43
44 BIN
45
46 \ t \ \ BINARY VARIABLES
47 \ t {}
48
49 END
50 """

```

```

52 def get_nodes(x, y, z):
53     """ Returns a tuple containing the source, transit and destination node
54     ids as integers. """
55     s = list(range(1, x + 1))
56     t = list(range(1, y + 1))
57     d = list(range(1, z + 1))
58     return s, t, d
59
60 def get_demand_constraints(s, t, d):
61     """ Returns a list of demand constraints. """
62     return [' + '.join(["x-{}{}{}2".format(i, k, j) for k in t]) + ' =
63     {}'.format(DEMANDFLOW(i, j))
64             for (i, j) in perms([s, d])]
65
66 def get_source_transit_capacity_constraints(s, t, d):
67     """ Returns a list of capacity constraints for the links between the
68     source and transit nodes. """
69     return \
70         [' + '.join(["x-{}{}{}2".format(i, k, j) for j in d]) +
71         ' - c-{}{}{}1 = 0'.format(i, k) for (i, k) in perms([s, t])]
72
73 def get_transit_destination_capacity_constraints(s, t, d):
74     """ Returns a list of capacity constraints for the links between the
75     transit and destination nodes. """
76     return \
77         [' + '.join(["x-{}{}{}2".format(i, k, j) for i in s]) +
78         ' - d-{}{}{}1 = 0'.format(k, j) for (k, j) in perms([t, d])]
79
80 def get_transit_load_constraints(s, t, d):
81     """ Returns the list of transit load constraints. """
82     return [' + '.join(["x-{}{}{}2".format(i, k, j) for (i, j) in perms([
83     s, d])]) +
84             ' - l-{} = 0'.format(k) for k in t]
85
86 def get_objective_function_load_constraints(s, t, d):
87     """ Returns the list of objective function load constraints. """
88     return [' + '.join(["c-{}{}{}1".format(i, j) for i in s]) +
89             ' - r <= 0' for j in d]
90
91 def get_binary_and_decision_variable_constraints(s, t, d):
92     """ Returns the binary and decision variable constraints. """
93     return ['{3} x-{}{}{}2 - {4} u-{}{}{}2 = 0'.format(i, k, j,
94     PATH_SPLIT, DEMANDFLOW(i, j)) for (i, k, j) in perms([s, t, d])]
95
96 def get_binary_constraints(s, t, d):
97     """ Returns a list of binary variable constraints. """
98     return [' + '.join(["u-{}{}{}2".format(i, k, j) for k in t]) + ' = {}
99     '.format(PATH_SPLIT)
100             for (i, j) in perms([s, d])]
101
102 def get_binary_variables(s, t, d):

```



```

104     """ Returns a list of binary variables. """
105     return ["u-{{0}}{{1}}{{2}}".format(i, k, j) for (i, k, j) in perms([s, t, d])
106 ]
107
108 def get_non_negativity_constraints(s, t, d):
109     """ Returns a list of non-negativity constraints. """
110     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])]
111
112 def generate_lp_file(title, authors, x, y, z):
113     """ Returns the LP file contents as per the project specification. """
114     s, t, d = get_nodes(x, y, z)
115
116     demand_constraints = get_lines(get_demand_constraints(s, t, d))
117     source_transit_capacity_constraints = get_lines(
118         get_source_transit_capacity_constraints(s, t, d))
119     transit_destination_capacity_constraints = get_lines(
120         get_transit_destination_capacity_constraints(s, t, d))
121     non_negativity_constraints = get_lines(get_non_negativity_constraints(
122         s, t, d))
123     objective_function_load_constraints = get_lines(
124         get_objective_function_load_constraints(s, t, d))
125     transit_load_constraints = get_lines(
126         get_transit_load_constraints(s, t, d))
127     binary_and_decision_constraints = get_lines(
128         get_binary_and_decision_variable_constraints(s, t, d))
129     binary_variable_constraints = get_lines(get_binary_constraints(s, t, d))
130     binary_variables = get_lines(get_binary_variables(s, t, d))
131
132     return TEMPLATE.format(
133         title,
134         authors,
135         x,
136         y,
137         z,
138         PATH_SPLIT,
139         get_function_source(DEMANDFLOW),
140         demand_constraints,
141         source_transit_capacity_constraints,
142         transit_destination_capacity_constraints,
143         objective_function_load_constraints,
144         transit_load_constraints,
145         binary_and_decision_constraints,
146         binary_variable_constraints,
147         non_negativity_constraints,
148         binary_variables)

```

../src/lp-gen.py

3.2 Generated LP File

3.2.1 problem_3_2_4.lp

\ COSC-364 Assignment 2, LP Output File

```

2 \ Written by Will Cowper (81163265), Jesse Sheehan (53366509)
3 \ Parameters: X=3, Y=2, Z=4, Split=2, Demand=2 * i + j
4
6 MINIMIZE
7     r
8
9 SUBJECT TO
10
11     \ DEMAND CONSTRAINTS
12     x_111 + x_121 = 3
13     x_112 + x_122 = 4
14     x_113 + x_123 = 5
15     x_114 + x_124 = 6
16     x_211 + x_221 = 5
17     x_212 + x_222 = 6
18     x_213 + x_223 = 7
19     x_214 + x_224 = 8
20     x_311 + x_321 = 7
21     x_312 + x_322 = 8
22     x_313 + x_323 = 9
23     x_314 + x_324 = 10
24
25     \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN SOURCE AND TRANSIT NODES
26     x_111 + x_112 + x_113 + x_114 - c_11 = 0
27     x_121 + x_122 + x_123 + x_124 - c_12 = 0
28     x_211 + x_212 + x_213 + x_214 - c_21 = 0
29     x_221 + x_222 + x_223 + x_224 - c_22 = 0
30     x_311 + x_312 + x_313 + x_314 - c_31 = 0
31     x_321 + x_322 + x_323 + x_324 - c_32 = 0
32
33     \ CAPACITY CONSTRAINTS FOR LINKS BETWEEN TRANSIT AND DESTINATION NODES
34     x_111 + x_211 + x_311 - d_11 = 0
35     x_112 + x_212 + x_312 - d_12 = 0
36     x_113 + x_213 + x_313 - d_13 = 0
37     x_114 + x_214 + x_314 - d_14 = 0
38     x_121 + x_221 + x_321 - d_21 = 0
39     x_122 + x_222 + x_322 - d_22 = 0
40     x_123 + x_223 + x_323 - d_23 = 0
41     x_124 + x_224 + x_324 - d_24 = 0
42
43     \ OBJECTIVE FUNCTION LOAD CONSTRAINTS
44     c_11 + c_21 + c_31 - r <= 0
45     c_12 + c_22 + c_32 - r <= 0
46     c_13 + c_23 + c_33 - r <= 0
47     c_14 + c_24 + c_34 - r <= 0
48
49     \ 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
51     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
53     \ BINARY VARIABLE AND DECISION VARIABLE CONSTRAINTS
54     2 x_111 - 3 u_111 = 0
55     2 x_112 - 4 u_112 = 0
56     2 x_113 - 5 u_113 = 0
57     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
92
BOUNDS
94
\ NON-NEGATIVITY CONSTRAINTS
96 r >= 0
    x_111 >= 0
98 x_112 >= 0
    x_113 >= 0
100 x_114 >= 0
    x_121 >= 0
102 x_122 >= 0
    x_123 >= 0
104 x_124 >= 0
    x_211 >= 0
106 x_212 >= 0
    x_213 >= 0
108 x_214 >= 0
    x_221 >= 0
110 x_222 >= 0
    x_223 >= 0
112 x_224 >= 0
    x_311 >= 0
114 x_312 >= 0
    x_313 >= 0

```

```
116  x_314 >= 0
      x_321 >= 0
118  x_322 >= 0
      x_323 >= 0
120  x_324 >= 0
      c_11 >= 0
122  c_12 >= 0
      c_21 >= 0
124  c_22 >= 0
      c_31 >= 0
126  c_32 >= 0
      d_11 >= 0
128  d_12 >= 0
      d_13 >= 0
130  d_14 >= 0
      d_21 >= 0
132  d_22 >= 0
      d_23 >= 0
134  d_24 >= 0

136  BIN
138  \  BINARY VARIABLES
      u_111
140  u_112
      u_113
142  u_114
      u_121
144  u_122
      u_123
146  u_124
      u_211
148  u_212
      u_213
150  u_214
      u_221
152  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
```

../problem_3.2.4.lp

3.3 Plagiarism Declaration