

COSC-364 FLOW PLANNING ASSIGNMENT

Will Cowper

ID: 81163265

wgc22@uclive.ac.nz

Contribution: 50%

Jesse Sheehan

ID: 53366509

jps111@uclive.ac.nz

Contribution: 50%

May 29, 2019

1 Problem Formulation

TALK ABOUT PROBLEM HERE. REFERENCE THE CONSTRAINT EQUATIONS AS WE GO. TRY TO ADDRESS EACH CONSTRAINT

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

$$\text{minimize}_{[x,c,d,r]} r \tag{1}$$

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

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

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

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

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

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__, ', '.join(
12         [" {0} ({1})".format(name, sid) for (name, sid) in __AUTHORS__]))

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

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

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

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

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

40
42 return x, y, z

44 def save_lp_file(filename, data):
    try:
46         f = open(filename, 'w')
        f.write(data)
48         f.close()
    except:
```

```

50     print("Error: could not save file '{0}'".format(filename))
51     exit(-1)
52
53
54 def main():
55     print_version()
56     if len(sys.argv) != 4:
57         print_usage()
58         exit(-1)
59     else:
60         x, y, z = get_problem_parameters()
61         data = generate_lp_file(x, y, z)
62         filename = get_lp_filename(x, y, z)
63         save_lp_file(filename, data)
64         print("Success: saved as '{0}'".format(filename))
65         run_cplex(filename)
66
67
68 if __name__ == "__main__":
69     main()

```

../src/__main__.py

3.1.2 src/lp_utils.py

```

import functools
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)
9
10
11 def run_cplex(filename):
12     """ Runs cplex on the LP file. """
13     subprocess.run(
14         ['cplex', '-c', "read {0}".format(filename), "optimize", "display solution variables -"])
15
16
17 def crange(first, last):
18     """ Returns a list of characters between the two characters passed in (inclusive).
19     >>> crange('A', 'C')
20     ['A', 'B', 'C']
21     >>> crange('A', 'A')
22     ['A']
23     """
24     if ord(first) > ord(last):
25         raise ValueError("last must come after first")
26
27     else:
28         return [chr(i) for i in range(ord(first), ord(last) + 1)]
29
30

```

```

def repeat(obj, n):
    """ Returns a list with obj repeated n times.
    >>> repeat(1, 1)
    [1]
    >>> repeat(42, 0)
    []
    >>> repeat(5, 4)
    [5, 5, 5, 5]
    >>> repeat([1, 2], 2)
    [[1, 2], [1, 2]]
    """
    return [obj for _ in range(n)]

def perms(lists):
    """ Returns all the permutations of the elements.
    >>> perms([])
    []
    >>> perms(['a', 'b', 'c'])
    [('a',), ('b',), ('c',)]
    >>> perms(['a', 'b', 'c'], ['x', 'y', 'z'])
    [('a', 'x'), ('a', 'y'), ('a', 'z'), ('b', 'x'), ('b', 'y'), ('b', 'z'),
    ('c', 'x'), ('c', 'y'), ('c', 'z')]
    """
    if len(lists) == 0:
        return []

    elif (len(lists) == 1):
        return [(x,) for x in lists[0]]

    else:
        return [(x,) + y for x in lists[0] for y in perms(lists[1:])]

def concat(permutations):
    """ Returns the permutations concatenated as strings.
    >>> concat(perms(['a', 'b', 'c']))
    ['a', 'b', 'c']
    >>> concat(perms(['a', 'b', 'c'], ['x', 'y', 'z']))
    ['ax', 'ay', 'az', 'bx', 'by', 'bz', 'cx', 'cy', 'cz']
    """
    return [functools.reduce(lambda x, y: x + str(y), p, '') for p in
    permutations]

def get_function_source(fn):
    src = inspect.getsource(fn)
    return src[src.index(':')+2:]

def get_lines(strings):
    return '\n\t'.join(strings)

if __name__ == "__main__":
    import doctest
    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 \\\ COSC-364 Assignment 2, LP Output File
9 \\\ Parameters: X={}, Y={}, Z={}, Split={}, Demand={}
10
11 MINIMIZE
12 \tr
13
14 SUBJECT TO
15
16 \t\\ DEMAND CONSTRAINTS
17 \t{}
18
19 \t\\ CAPACITY CONSTRAINTS FOR LINKS BETWEEN SOURCE AND TRANSIT NODES
20 \t{}
21
22 \t\\ CAPACITY CONSTRAINTS FOR LINKS BETWEEN TRANSIT AND DESTINATION NODES
23 \t{}
24
25 \t\\ OBJECTIVE FUNCTION LOAD CONSTRAINTS
26 \t{}
27
28 \t\\ TRANSIT NODE LOAD CONSTRAINTS
29 \t{}
30
31 \t\\ BINARY VARIABLE AND DECISION VARIABLE CONSTRAINTS
32 \t{}
33
34 \t\\ BINARY VARIABLE CONSTRAINTS (ONLY 2 ACTIVE TRANSIT NODES)
35 \t{}
36
37 BOUNDS
38
39 \t\\ NON-NEGATIVITY CONSTRAINTS
40 \tr >= 0
41 \t{}
42
43 BIN
44
45 \t\\ BINARY VARIABLES
46 \t{}
47
48 END
49 """
50
51
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))

```

```

56     d = list(range(1, z + 1))
57     return s, t, d
58
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):
103     """ Returns a list of binary variables. """
104     return ["u-{}{}{}2".format(i, k, j) for (i, k, j) in perms([s, t, d])]
105
106 def get_non_negativity_constraints(s, t, d):

```



```

108     """ Returns a list of non-negativity constraints. """
109     return ["x-{{0}}{{1}}{{2}} >= 0".format(i, k, j) for (i, k, j) in perms([s, t
110 , d])] + ["c-{{0}}{{1}} >= 0".format(i, k) for (i, k) in perms([s, t])] + ["
111 d-{{0}}{{1}} >= 0".format(k, j) for (k, j) in perms([t, d])]
112
113 def generate_lp_file(x, y, z):
114     """ Returns the LP file contents as per the project specification. """
115     s, t, d = get_nodes(x, y, z)
116
117     demand_constraints = get_lines(get_demand_constraints(s, t, d))
118     source_transit_capacity_constraints = get_lines(
119         get_source_transit_capacity_constraints(s, t, d))
120     transit_destination_capacity_constraints = get_lines(
121         get_transit_destination_capacity_constraints(s, t, d))
122     non_negativity_constraints = get_lines(get_non_negativity_constraints(
123         s, t, d))
124     objective_function_load_constraints = get_lines(
125         get_objective_function_load_constraints(s, t, d))
126     transit_load_constraints = get_lines(
127         get_transit_load_constraints(s, t, d))
128     binary_and_decision_constraints = get_lines(
129         get_binary_and_decision_variable_constraints(s, t, d))
130     binary_variable_constraints = get_lines(get_binary_constraints(s, t, d)
131 )
132     binary_variables = get_lines(get_binary_variables(s, t, d))
133
134     return TEMPLATE.format(
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

3.3 Plagiarism Declaration