COSC 364 Assignment 2

31 May 2017

|  |  |
| --- | --- |
| **Noel Soong**  56244036  50% | **Alexis Sy**  21229382  50% |

Minimize

(The objective function is to minimize r, for the load to be balanced)

Subject to

(This is the demand constraint so that the sum of all decision variables meets the demand needed for a source going to a destination)

(Both lines above are the capacity constraints, so that the decision variables do not exceed the capacity on the links required for the source to get to the destination)

(for decision variables coming/going to/from a transit node, it has to be less than or equal to r, to be considered to be load balanced)

(To be split into three different paths)

(if decision variable is used the load must be divided by 3 for equal split to be possible)

Bounds

(the above are non-negativity constraints)

(indicates whether the path is used or not)

|  |
| --- |
| LP file generated for X = 3, Y = 2, Z = 4 |
| Minimize  r  Subject to  x111 + x121 = 2  x112 + x122 = 3  x113 + x123 = 4  x114 + x124 = 5  x211 + x221 = 3  x212 + x222 = 4  x213 + x223 = 5  x214 + x224 = 6  x311 + x321 = 4  x312 + x322 = 5  x313 + x323 = 6  x314 + x324 = 7  x111 + x112 + x113 + x114 - c11 <= 0  x121 + x122 + x123 + x124 - c12 <= 0  x211 + x212 + x213 + x214 - c21 <= 0  x221 + x222 + x223 + x224 - c22 <= 0  x311 + x312 + x313 + x314 - c31 <= 0  x321 + x322 + x323 + x324 - c32 <= 0  x111 + x211 + x311 - d11 <= 0  x121 + x221 + x321 - d21 <= 0  x112 + x212 + x312 - d12 <= 0  x122 + x222 + x322 - d22 <= 0  x113 + x213 + x313 - d13 <= 0  x123 + x223 + x323 - d23 <= 0  x114 + x214 + x314 - d14 <= 0  x124 + x224 + x324 - d24 <= 0  x111 + x112 + x113 + x114 + x211 + x212 + x213 + x214 + x311 + x312 + x313 + x314 - r <= 0  x121 + x122 + x123 + x124 + x221 + x222 + x223 + x224 + x321 + x322 + x323 + x324 - r <= 0  u111 + u121 = 3  u112 + u122 = 3  u113 + u123 = 3  u114 + u124 = 3  u211 + u221 = 3  u212 + u222 = 3  u213 + u223 = 3  u214 + u224 = 3  u311 + u321 = 3  u312 + u322 = 3  u313 + u323 = 3  u314 + u324 = 3  3 x111 - 2 u111 = 0  3 x121 - 2 u121 = 0  3 x112 - 3 u112 = 0  3 x122 - 3 u122 = 0  3 x113 - 4 u113 = 0  3 x123 - 4 u123 = 0  3 x114 - 5 u114 = 0  3 x124 - 5 u124 = 0  3 x211 - 3 u211 = 0  3 x221 - 3 u221 = 0  3 x212 - 4 u212 = 0  3 x222 - 4 u222 = 0  3 x213 - 5 u213 = 0  3 x223 - 5 u223 = 0  3 x214 - 6 u214 = 0  3 x224 - 6 u224 = 0  3 x311 - 4 u311 = 0  3 x321 - 4 u321 = 0  3 x312 - 5 u312 = 0  3 x322 - 5 u322 = 0  3 x313 - 6 u313 = 0  3 x323 - 6 u323 = 0  3 x314 - 7 u314 = 0  3 x324 - 7 u324 = 0  Bounds  r >= 0  x111 >= 0  x121 >= 0  x112 >= 0  x122 >= 0  x113 >= 0  x123 >= 0  x114 >= 0  x124 >= 0  x211 >= 0  x221 >= 0  x212 >= 0  x222 >= 0  x213 >= 0  x223 >= 0  x214 >= 0  x224 >= 0  x311 >= 0  x321 >= 0  x312 >= 0  x322 >= 0  x313 >= 0  x323 >= 0  x314 >= 0  x324 >= 0  c11 >= 0  c12 >= 0  c21 >= 0  c22 >= 0  c31 >= 0  c32 >= 0  d11 >= 0  d12 >= 0  d13 >= 0  d14 >= 0  d21 >= 0  d22 >= 0  d23 >= 0  d24 >= 0  Binary  u111  u112  u113  u114  u121  u122  u123  u124  u211  u212  u213  u214  u221  u222  u223  u224  u311  u312  u313  u314  u321  u322  u323  u324  End |

|  |
| --- |
| Source code for the program |
| import re  import subprocess  from sys import \*  import time  SOURCE\_LEN = None  TRANSIT\_LEN = None  DEST\_LEN = None  """Main loop takes in number for source,transit and destination nodes"""  while True:  try:  x = input("Input Number of SOURCE Nodes: ")  if int(x) >= 1:  SOURCE\_LEN = int(x)  else:  exit("NUMBER SHOULD BE GREATER OR EQUALS 1")    x = input("Input Number of TRANSIT Nodes: ")  if int(x) >= 1:  TRANSIT\_LEN = int(x)  else:  exit("NUMBER SHOULD BE GREATER OR EQUALS 1")    x = input("Input Number of DESTINATION Nodes: ")  if int(x) >= 1:  DEST\_LEN = int(x)  else:  exit("NUMBER SHOULD BE GREATER OR EQUALS 1")    if SOURCE\_LEN and TRANSIT\_LEN and DEST\_LEN:  break  except SyntaxError as ans:  exit(str(ans))      SOURCE = list(range(1,SOURCE\_LEN+1))  TRANS = list(range(1,TRANSIT\_LEN+1))  DEST = list(range(1,DEST\_LEN+1))  demand\_table = dict()  CACHE = []  """Create demand\_table of demand volumes from S to D"""  for s in SOURCE:  demand\_table[s] = []  for d in DEST:  result = s + d  demand\_table[s].append(result)  print(demand\_table)    stringy = open("loadbalance.lp", "w")  content = """Minimize  r\n"""  content += """Subject to \n"""  #Demand Constraints  for i,s in enumerate(SOURCE):  for j,d in enumerate(DEST):  for k,tn in enumerate(TRANS):  decision\_var = "x" + str(s) + str(tn) + str(d)  CACHE.append(decision\_var)  CACHE.append(demand\_table[s][j])  line = ' {} +' \* (len(TRANS)-1)  line += ' {} = {}\n'  content += line.format(\*CACHE)  CACHE = []  #Capacity Constraints  for i,s in enumerate(SOURCE):  for k,tn in enumerate(TRANS):  capacity\_var = "c" + str(s) + str(tn)  for j,d in enumerate(DEST):  decision\_var = "x" + str(s) + str(tn) + str(d)  CACHE.append(decision\_var)  CACHE.append(capacity\_var)  line = " {} +" \* (len(DEST)-1)  line += ' {} - {} <= 0\n'  content += line.format(\*CACHE)  CACHE = []    #Capacity Constraints  for j,d in enumerate(DEST):  for k,tn in enumerate(TRANS):  capacity\_var = "d" + str(tn) + str(d)  for i,s in enumerate(SOURCE):  decision\_var = "x" + str(s) + str(tn) + str(d)  CACHE.append(decision\_var)  CACHE.append(capacity\_var)  line = " {} +" \* (len(SOURCE)-1)  line += ' {} - {} <= 0\n'  content += line.format(\*CACHE)  CACHE = []  #Load Balancing  for k,tn in enumerate(TRANS):  for i,s in enumerate(SOURCE):  for j,d in enumerate(DEST):  decision\_var = "x" + str(s) + str(tn) + str(d)  CACHE.append(decision\_var)  CACHE.append("r")  line = " {} +" \* (len(SOURCE)\*len(DEST)-1)  line += " {} - r <= 0\n"  content += line.format(\*CACHE)  CACHE = []    for i,s in enumerate(SOURCE):  for j,d in enumerate(DEST):  for k,tn in enumerate(TRANS):  var = "u" + str(s) + str(tn) + str(d)  CACHE.append(var)  line = " {} +" \* (len(TRANS)-1)  line += " {} = 3\n"  content += line.format(\*CACHE)  CACHE = []  for i,s in enumerate(SOURCE):  for j,d in enumerate(DEST):  for k,tn in enumerate(TRANS):  decision\_var = "x" + str(s) + str(tn) + str(d)  var = "u" + str(s) + str(tn) + str(d)  CACHE.append(decision\_var)  CACHE.append(demand\_table[s][j])  CACHE.append(var)  content += " 3 {} - {} {} = 0\n".format(\*CACHE)  CACHE = []      content += "Bounds\n"  content += " r >= 0\n"  for i,s in enumerate(SOURCE):  for j,d in enumerate(DEST):  for k,tn in enumerate(TRANS):  decision\_var = "x" + str(s) + str(tn) + str(d)  content += " {} >= 0\n".format(decision\_var)    for i,s in enumerate(SOURCE):  for k,tn in enumerate(TRANS):  capacity\_var = "c" + str(s) + str(tn)  content += " {} >= 0\n".format(capacity\_var)    for k,tn in enumerate(TRANS):  for j,d in enumerate(DEST):  capacity\_var = "d" + str(tn) + str(d)  content += " {} >= 0\n".format(capacity\_var)  content += "Binary\n"  for i,s in enumerate(SOURCE):  for k,tn in enumerate(TRANS):  for j,d in enumerate(DEST):  var = "u" + str(s) + str(tn) + str(d)  content += " {}\n".format(var)  content += """End\n"""  stringy.write(content)  stringy.close()  start = time.time()  p = subprocess.Popen('cplex -c \"read loadbalance.lp\" \"optimize\" \"display solution variables -"', stdout=subprocess.PIPE, shell=True)  stop = time.time()  table = dict()  maxCap = 0  maxCapList = set()  numLinks = 0  for line in p.stdout.readlines():  ln = line.split()  if line[0][0] == "c" or line[0][0] == "d":  numLinks += 1  if ln:  if ln[0][0] == "r":  print("\nr = "+ str(ln[1]) + "\n")  if ln[0][0] == "x":  if ln[0][2] not in table:  table[ln[0][2]] = round(float(ln[1]), 3)  else:  table[ln[0][2]] += round(float(ln[1]),3)  if (ln[0][0] == "c" or ln[0][0] == "d") and ln[1] > maxCap:  maxCap = ln[1]  maxCapList = set()  maxCapList.add(ln[0])  if (ln[0][0] == "c" or ln[0][0] == "d") and ln[1] == maxCap:  maxCapList.add(ln[0])  print("\nNumber of non-zero links: "+ str(numLinks))  print("\nMAX CAP")  for cp in maxCapList:  print(" " + str(cp)+":" + str(maxCap))  print("\nTransit Loads")  print(table)  print("Total Time Taken: "+ str((stop - start)/60)+"s") |

7-3-7

Time taken : 1.5648206075e-05s

Max Capacity links:

d17:26.000000  
 c71:26.000000  
 d37:26.000000  
 c73:26.000000  
 c72:26.000000  
 d27:26.000000  
Sum Transit Loads:

1 : 130.666667

2: 130.666667

3: 130.666667  
Number of None-Zero Links:

42

7-4-7

R = 98.000000

Time taken : 1.23500823975e-05s

Max Capacity links:

c23:9.000000  
 d23:9.000000  
 d47:9.000000  
 d11:9.000000  
 c32:9.000000  
Sum Transit Loads:

1 : 98.000000

2: 98.000000

3: 98.000000

4: 98.000000  
Number of non-zero links: 56

7-5-7

R = 78.666667

Time taken : 1.48812929789e-05s

Max Capacity links:

d14:9.000000  
 d42:9.000000  
 c11:9.000000

c21:9.000000  
 c14:9.000000  
  
Sum Transit Loads:

1 : 78.666667

2: 78.666667

3: 78.666667

4: 78.666667

5: 78.666667  
Number of non-zero links: 68

7-6-7

R = 65.333333

Time taken : 1.4583269755e-05s

Max Capacity links:

c45:9.000000  
 c66:9.000000  
 c31:9.000000  
 c76:9.000000  
 c11:9.000000  
 d54:9.000000  
Sum Transit Loads:

1 : 78.666667

2: 78.666667

3: 78.666667

4: 78.666667

5: 78.666667

6: 78.666667

Number of non-zero links: 80

7-6-7

R = 56.000000

Time taken : 1.46349271139e-05s

Max Capacity links:

d15:9.000000  
 c46:9.000000  
 d71:9.000000  
 c27:9.000000  
 c51:9.000000  
 c43:9.000000  
 c34:9.000000  
 c11:9.000000  
 d12:9.000000  
Sum Transit Loads:

1 : 56.000000

2: 56.000000

3: 56.000000

4: 56.000000

5: 56.000000

6: 56.000000

7: 56.000000  
Number of non-zero links: 92