**CEE 598: Traffic Simulation Modeling and Applications, 2020**

**Take Home Final exam, Sunday, 5PM**

**Student Name: Adam Tran**

1. Provide pseudo codes for estimating π using Monte Carlo simulation. What are expected standard deviation and 95% confidence level of the π estimate as a function of sample size?

**Total: 15 points**

Pseudo code:

|  |
| --- |
| int x[100], int y[100]; % Number of points, can be of any n x n size  for (i=0; i<100,i++)  { x[i] = uniform [0,1] % x and y of any random decimal from 0 to 1  y[i] = uniform [0,1] }  int count = 0;  if (x\*x + y\*y <= 1) % if statement that checks if points are within bounds  then count = count + 1;  float pie = 4\*count/100 % estimation of π |

Sample standard deviation is given by the following equation:

where is the sample mean

is the sample size

The 95% confidence interval on the sample mean is given by the following equation:

Where is the Z-value for the confidence interval (95%)

is the sample standard deviation

1. Assume there are two factories in Memphis and Denver producing 150K and 200K of a product per day, respectively. We also have customers in Los Angeles and Boston, with a demand of 175K per day each. The Management and decision makers believe that it may be cheaper to first ship part of the product to New York or Chicago and then ship them to their final destinations. The costs of shipping a 1K batch of product are given in the table below. Management wants to minimize the total cost of shipping. Provide a mathematical formulation for this problem **(10 points)** and then solve it by GAMS **(15 points)** and Excel solver both **(10 points)**. (Tips: this is a general transshipment problem).

**Total: 35 points**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| To ($) | | | | | | |
| From | Memphis | Denver | N.Y. | Chicago | L.A. | Boston |
| Memphis | 0 | - | 8 | 13 | 25 | 28 |
| Denver | - | 0 | 15 | 12 | 26 | 25 |
| N.Y. | - | - | 0 | 6 | 16 | 17 |
| Chicago | - | - | 6 | 0 | 14 | 16 |
| L.A. | - | - | - | - | 0 | - |
| Boston | - | - | - | - | - | 0 |

Mathematical formulation:

: decision variable

Minimize (objective function)

Subject to: (structural/ technological constraints)

(nonnegativity constraints)

Minimize

Subject to

for all

,

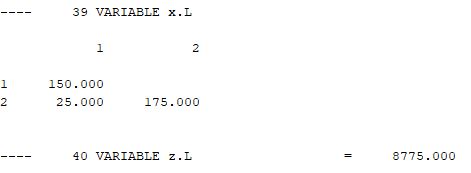
GAMS Solution:

Input Data

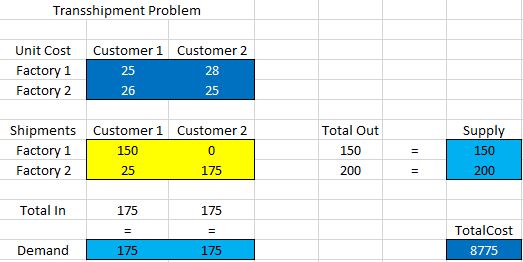
|  |
| --- |
| $title Transportation problem  Set i /1\*2/;  alias (i, j);  parameter a(i)/  1 150  2 200  /;  parameter b(j)/  1 175  2 175  /;  parameter c(i,j)/  1. 1 25  1. 2 28  2. 1 26  2. 2 25  /;  variable z;  positive variables x(i,j);  equations  obj  demand(i)  supply(j);  obj.. z =e= sum((i,j),c(i,j)\*x(i,j));  demand(i).. sum(j,x(i,j)) =e= a(i);  supply(j).. sum(i,x(i,j)) =e= b(j);  Model problem\_2 /all/;  solve problem\_2 using LP minimizing z;  display x.l;  display z.l; |

Output Data:

|  |
| --- |
| GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 1  Transportation problem  C o m p i l a t i o n  2  3 Set i /1\*2/;  4 alias (i, j);  5  6 parameter a(i)/  7 1 150  8 2 200  9 /;  10  11 parameter b(j)/  12 1 175  13 2 175  14 /;  15  16 parameter c(i,j)/  17 1. 1 25  18 1. 2 28  19 2. 1 26  20 2. 2 25  21 /;  22  23 variable z;  24 positive variables x(i,j);  25  26 equations  27 obj  28 demand(i)  29 supply(j)  30 ;  31  32 obj.. z =e= sum((i,j),c(i,j)\*x(i,j));  33 demand(i).. sum(j,x(i,j)) =e= a(i);  34 supply(j).. sum(i,x(i,j)) =e= b(j);  35  36 Model problem\_2 /all/;  37 solve problem\_2 using LP minimizing z;  38  39 display x.l;  40 display z.l;  COMPILATION TIME = 0.000 SECONDS 3 MB 32.2.0 rc62c018 WEX-WEI  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 2  Transportation problem  Equation Listing SOLVE problem\_2 Using LP From line 37  ---- obj =E=  obj.. z - 25\*x(1,1) - 28\*x(1,2) - 26\*x(2,1) - 25\*x(2,2) =E= 0 ; (LHS = 0)    ---- demand =E=  demand(1).. x(1,1) + x(1,2) =E= 150 ; (LHS = 0, INFES = 150 \*\*\*\*)    demand(2).. x(2,1) + x(2,2) =E= 200 ; (LHS = 0, INFES = 200 \*\*\*\*)    ---- supply =E=  supply(1).. x(1,1) + x(2,1) =E= 175 ; (LHS = 0, INFES = 175 \*\*\*\*)    supply(2).. x(1,2) + x(2,2) =E= 175 ; (LHS = 0, INFES = 175 \*\*\*\*)    GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 3  Transportation problem  Column Listing SOLVE problem\_2 Using LP From line 37  ---- z  z  (.LO, .L, .UP, .M = -INF, 0, +INF, 0)  1 obj  ---- x  x(1,1)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -25 obj  1 demand(1)  1 supply(1)  x(1,2)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -28 obj  1 demand(1)  1 supply(2)  x(2,1)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -26 obj  1 demand(2)  1 supply(1)  REMAINING ENTRY SKIPPED  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 4  Transportation problem  Model Statistics SOLVE problem\_2 Using LP From line 37  MODEL STATISTICS  BLOCKS OF EQUATIONS 3 SINGLE EQUATIONS 5  BLOCKS OF VARIABLES 2 SINGLE VARIABLES 5  NON ZERO ELEMENTS 13  GENERATION TIME = 1.266 SECONDS 4 MB 32.2.0 rc62c018 WEX-WEI  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 5  Transportation problem  Solution Report SOLVE problem\_2 Using LP From line 37  S O L V E S U M M A R Y  MODEL problem\_2 OBJECTIVE z  TYPE LP DIRECTION MINIMIZE  SOLVER CPLEX FROM LINE 37  \*\*\*\* SOLVER STATUS 1 Normal Completion  \*\*\*\* MODEL STATUS 1 Optimal  \*\*\*\* OBJECTIVE VALUE 8775.0000  RESOURCE USAGE, LIMIT 0.031 10000000000.000  ITERATION COUNT, LIMIT 0 2147483647  IBM ILOG CPLEX 32.2.0 rc62c018 Released Aug 26, 2020 WEI x86 64bit/MS Window  \*\*\* This solver runs with a demo license. No commercial use.  Cplex 12.10.0.0  Space for names approximately 0.00 Mb  Use option 'names no' to turn use of names off  LP status(1): optimal  Cplex Time: 0.00sec (det. 0.00 ticks)  Optimal solution found.  Objective : 8775.000000  LOWER LEVEL UPPER MARGINAL  ---- EQU obj . . . 1.000  ---- EQU demand  LOWER LEVEL UPPER MARGINAL  1 150.000 150.000 150.000 24.000  2 200.000 200.000 200.000 25.000  ---- EQU supply  LOWER LEVEL UPPER MARGINAL  1 175.000 175.000 175.000 1.000  2 175.000 175.000 175.000 .  LOWER LEVEL UPPER MARGINAL  ---- VAR z -INF 8775.000 +INF .  ---- VAR x  LOWER LEVEL UPPER MARGINAL  1.1 . 150.000 +INF .  1.2 . . +INF 4.000  2.1 . 25.000 +INF .  2.2 . 175.000 +INF .  \*\*\*\* REPORT SUMMARY : 0 NONOPT  0 INFEASIBLE  0 UNBOUNDED  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/05/20 15:49:57 Page 6  Transportation problem  E x e c u t i o n  ---- 39 VARIABLE x.L  1 2  1 150.000  2 25.000 175.000  ---- 40 VARIABLE z.L = 8775.000  EXECUTION TIME = 3.828 SECONDS 4 MB 32.2.0 rc62c018 WEX-WEI  USER: GAMS Demo license for Dr. Zhou G200810|0002CO-GEN  Arizona State Univeristy, United States of America DL014724  \*\*\*\* FILE SUMMARY  Input C:\Users\Adam Tran\Documents\CEE598 \_TrafficSimulation\Final\Simulati  onSpreadsheets\1\_Transportation problem.gms  Output C:\Users\Adam Tran\Documents\gamsdir\projdir\1\_Transportation problem  .lst |



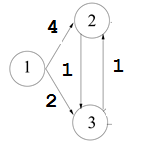
Excel Solution:



To create a network with intermediate nodes, constraints on the production and attractions ends need to be defined further. Solving the problem can then be done as a generalized network flow problem with constraints.

1. For the following network:
2. Show the corresponding node-arc incidence matrix and adjacency list **(5 points).**
3. Provide pseudo codes for the label correcting-based shortest path algorithm **(5 points)**.
4. Define all parameters and variables clearly in both input and output data **(5 points)**.
5. Show the shortest paths from node 1 and nodes 2 and 3, respectively, represented in terms of node predecessors. Numbers on arcs are generalized costs **(10 points)**.

**Total: 25 points.**



1. Node-arc incidence matrix

Rows correspond to nodes, column to arcs. The column for arc has exactly two non-zero entries: +1 In row and -1 in row .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | (1,2) | (1,3) | (2,3) | (3,2) |
| 1 | 1 | 1 | 0 | 0 |
| 2 | -1 | 0 | 1 | -1 |
| 3 | 0 | -1 | -1 | 1 |

Adjacency list representation

* + 1. For each node , the arc adjacency list A(i) is stored as a linked list
    2. Each record in the linked list corresponds to an arc , and stores the following info:
       - 1. The head of the arc
         2. The cost
         3. A pointer to the next record in the linked list
    3. An array of pointers is used to stores a pointer to the first record of each linked list

1. Label Correcting Shortest-Path Algorithm

Initialize:

* + - 1. , % Start to end node
      2. % Shortest path distance
      3. Update , where
      4. Select node from
      5. Update make node permanently labeled
      6. Terminate when is empty or

Pseudo-Code Python

|  |
| --- |
| def optimal\_label\_correcting(self,origin\_node,destination\_node,departure\_time):  global \_MAX\_LABEL\_COST    if len(g\_node\_list[origin\_node].outgoing\_node\_list) == 0:  return 0    for i in range(g\_number\_of\_nodes): #Initialization for all nodes  self.node\_status\_array[i] = 0 #not scanned  self.node\_label\_cost[i] = \_MAX\_LABEL\_COST  self.node\_predecessor[i] = -1 #pointer to previous NODE INDEX from the current label at current node and time  self.link\_predecessor[i] = -1 #pointer to previous NODE INDEX from the current label at current link and time    self.node\_label\_cost[origin\_node] = departure\_time  SEList = []  SEList.append(origin\_node)    while len(SEList)>0:  from\_node = SEList[0]  del SEList[0]  # print('length of outgoing\_node\_dict of from\_node:',len(agent.outgoing\_node\_dict[from\_node]))  for k in range(len(g\_node\_list[from\_node].outgoing\_node\_list)):  to\_node = g\_node\_list[from\_node].outgoing\_node\_list[k].to\_node\_seq\_no  #print('to\_node:',to\_node)  b\_node\_updated = False  new\_to\_node\_cost = self.node\_label\_cost[from\_node] + self.link\_cost\_array[g\_node\_list[from\_node].outgoing\_node\_list[k].link\_seq\_no]  #print('new\_to\_node cost:',new\_to\_node\_cost,'from\_node\_label\_cost:',self.node\_label\_cost[from\_node],'to\_node\_label\_cost:',self.node\_label\_cost[to\_node])  if (new\_to\_node\_cost < self.node\_label\_cost[to\_node]): #we only compare cost at the downstream node ToID at the new arrival time t  # update cost label and node/time predecessor  self.node\_label\_cost[to\_node] = new\_to\_node\_cost  #print('new\_to\_node\_label\_cost:',self.node\_label\_cost[to\_node])  self.node\_predecessor[to\_node] = from\_node #pointer to previous physical NODE INDEX from the current label at current node and time  self.link\_predecessor[to\_node] = g\_node\_list[from\_node].outgoing\_node\_list[k].link\_seq\_no #pointer to previous physical NODE INDEX from the current label at current node and time  b\_node\_updated = True    SEList.append(to\_node)  #print('SEList after:',SEList)  self.node\_status\_array[to\_node] = 1  #print('from\_node:',from\_node,'to\_node:',to\_node,'from\_node\_label\_cost:',self.node\_label\_cost[from\_node],'to\_node\_label\_cost\_original:',self.node\_label\_cost[to\_node],'new\_to\_node cost:',new\_to\_node\_cost,)    if (destination\_node >= 0 and self.node\_label\_cost[destination\_node] < \_MAX\_LABEL\_COST):  return 1  elif (destination\_node == -1):  return 1 # one to all shortest path  else:  return -1 |

1. Input Data GAMS (4-Node Network)

|  |
| --- |
| $title Shortest Path Problem  \*LIMROW = 0, LIMCOL = 0  \*OPTIONS ITERLIM=100000, RESLIM = 1000000, SYSOUT = OFF, SOLPRINT = OFF, lp = COINGLPK, mip = COINGLPK, OPTCR= 0.1;  set i nodes /1\*3/;  alias (i, j);  parameter w(i,j) link travel time /  1. 2 2  1. 3 1  2. 4 1  3. 4 2  3. 2 1  /;  parameter origin(i);  origin('1') = 1;  parameter destination(i);  destination('4') = 4;  parameter intermediate\_node(i);  intermediate\_node(i) = (1- origin(i))\*(1- destination(i));  variable z;  positive variables  x(i,j) selection of flow between i and j;  equations  so\_obj define objective function  flow\_on\_node\_origin  flow\_on\_node\_intermediate(i)  flow\_on\_node\_destination  ;  so\_obj.. z =e= sum((i,j)$(w(i,j)),w(i,j)\*x(i,j));  flow\_on\_node\_origin.. sum(j$(w('1',j)), x('1',j)) =e= 1;  flow\_on\_node\_intermediate(i)$(intermediate\_node(i)=1).. sum(j$(w(i,j)), x(i,j))-sum(j$(w(j,i)), x(j,i))=e= 0;  flow\_on\_node\_destination.. sum(j$(w(j,'4')), x(j,'4'))=e= 1;  Model shortest\_path\_problem /all/ ;  solve shortest\_path\_problem using LP minimizing z;  display x.l;  display z.l; |

Output Data GAMS (4-Node Network)

|  |
| --- |
| GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 1  Shortest Path Problem  C o m p i l a t i o n  2 \*LIMROW = 0, LIMCOL = 0  3 \*OPTIONS ITERLIM=100000, RESLIM = 1000000, SYSOUT = OFF, SOLPRINT = OFF,  lp = COINGLPK, mip = COINGLPK, OPTCR= 0.1;  4  5 set i nodes /1\*4/;  6 alias (i, j);  7  8 parameter w(i,j) link travel time /  9 1. 2 2  10 1. 3 1  11 2. 4 1  12 3. 4 2  13 3. 2 1  14 /;  15  16 parameter origin(i);  17 origin('1') = 1;  18  19 parameter destination(i);  20 destination('4') = 4;  21  22 parameter intermediate\_node(i);  23 intermediate\_node(i) = (1- origin(i))\*(1- destination(i));  24  25 variable z;  26 positive variables  27 x(i,j) selection of flow between i and j;  28  29 equations  30 so\_obj define objective function  31 flow\_on\_node\_origin  32 flow\_on\_node\_intermediate(i)  33 flow\_on\_node\_destination  34 ;  35  36 so\_obj.. z =e= sum((i,j)$(w(i,j)),w(i,j)\*x(i,j));  37 flow\_on\_node\_origin.. sum(j$(w('1',j)), x('1',j)) =e= 1;  38 flow\_on\_node\_intermediate(i)$(intermediate\_node(i)=1).. sum(j$(w(i,j)), x(  i,j))-sum(j$(w(j,i)), x(j,i))=e= 0;  39 flow\_on\_node\_destination.. sum(j$(w(j,'4')), x(j,'4'))=e= 1;  40  41 Model shortest\_path\_problem /all/ ;  42  43 solve shortest\_path\_problem using LP minimizing z;  44  45 display x.l;  46 display z.l;  47  COMPILATION TIME = 0.000 SECONDS 3 MB 32.2.0 rc62c018 WEX-WEI  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 2  Shortest Path Problem  Equation Listing SOLVE shortest\_path\_problem Using LP From line 43  ---- so\_obj =E= define objective function  so\_obj.. z - 2\*x(1,2) - x(1,3) - x(2,4) - x(3,2) - 2\*x(3,4) =E= 0 ; (LHS = 0)    ---- flow\_on\_node\_origin =E=  flow\_on\_node\_origin.. x(1,2) + x(1,3) =E= 1 ; (LHS = 0, INFES = 1 \*\*\*\*)    ---- flow\_on\_node\_intermediate =E=  flow\_on\_node\_intermediate(2).. - x(1,2) + x(2,4) - x(3,2) =E= 0 ; (LHS = 0)    flow\_on\_node\_intermediate(3).. - x(1,3) + x(3,2) + x(3,4) =E= 0 ; (LHS = 0)    ---- flow\_on\_node\_destination =E=  flow\_on\_node\_destination.. x(2,4) + x(3,4) =E= 1 ; (LHS = 0, INFES = 1 \*\*\*\*)    GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 3  Shortest Path Problem  Column Listing SOLVE shortest\_path\_problem Using LP From line 43  ---- z  z  (.LO, .L, .UP, .M = -INF, 0, +INF, 0)  1 so\_obj  ---- x selection of flow between i and j  x(1,2)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -2 so\_obj  1 flow\_on\_node\_origin  -1 flow\_on\_node\_intermediate(2)  x(1,3)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -1 so\_obj  1 flow\_on\_node\_origin  -1 flow\_on\_node\_intermediate(3)  x(2,4)  (.LO, .L, .UP, .M = 0, 0, +INF, 0)  -1 so\_obj  1 flow\_on\_node\_intermediate(2)  1 flow\_on\_node\_destination  REMAINING 2 ENTRIES SKIPPED  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 4  Shortest Path Problem  Model Statistics SOLVE shortest\_path\_problem Using LP From line 43  MODEL STATISTICS  BLOCKS OF EQUATIONS 4 SINGLE EQUATIONS 5  BLOCKS OF VARIABLES 2 SINGLE VARIABLES 6  NON ZERO ELEMENTS 16  GENERATION TIME = 0.015 SECONDS 4 MB 32.2.0 rc62c018 WEX-WEI  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 5  Shortest Path Problem  Solution Report SOLVE shortest\_path\_problem Using LP From line 43  S O L V E S U M M A R Y  MODEL shortest\_path\_problem OBJECTIVE z  TYPE LP DIRECTION MINIMIZE  SOLVER CPLEX FROM LINE 43  \*\*\*\* SOLVER STATUS 1 Normal Completion  \*\*\*\* MODEL STATUS 1 Optimal  \*\*\*\* OBJECTIVE VALUE 3.0000  RESOURCE USAGE, LIMIT 0.000 10000000000.000  ITERATION COUNT, LIMIT 0 2147483647  IBM ILOG CPLEX 32.2.0 rc62c018 Released Aug 26, 2020 WEI x86 64bit/MS Window  \*\*\* This solver runs with a demo license. No commercial use.  Cplex 12.10.0.0  Space for names approximately 0.00 Mb  Use option 'names no' to turn use of names off  LP status(1): optimal  Cplex Time: 0.00sec (det. 0.01 ticks)  Optimal solution found.  Objective : 3.000000  LOWER LEVEL UPPER MARGINAL  ---- EQU so\_obj . . . 1.000  ---- EQU flow\_on\_n~ 1.000 1.000 1.000 1.000  so\_obj define objective function  ---- EQU flow\_on\_node\_intermediate  LOWER LEVEL UPPER MARGINAL  2 . . . -1.000  3 . . . .  LOWER LEVEL UPPER MARGINAL  ---- EQU flow\_on\_n~ 1.000 1.000 1.000 2.000  LOWER LEVEL UPPER MARGINAL  ---- VAR z -INF 3.000 +INF .  ---- VAR x selection of flow between i and j  LOWER LEVEL UPPER MARGINAL  1.2 . . +INF EPS  1.3 . 1.000 +INF .  2.4 . 1.000 +INF .  3.2 . 1.000 +INF .  3.4 . . +INF EPS  \*\*\*\* REPORT SUMMARY : 0 NONOPT  0 INFEASIBLE  0 UNBOUNDED  GAMS 32.2.0 rc62c018 Released Aug 26, 2020 WEX-WEI x86 64bit/MS Windows - 12/04/20 11:44:16 Page 6  Shortest Path Problem  E x e c u t i o n  ---- 45 VARIABLE x.L selection of flow between i and j  2 3 4  1 1.000  2 1.000  3 1.000  ---- 46 VARIABLE z.L = 3.000  EXECUTION TIME = 0.078 SECONDS 4 MB 32.2.0 rc62c018 WEX-WEI  USER: GAMS Demo license for Dr. Zhou G200810|0002CO-GEN  Arizona State Univeristy, United States of America DL014724  \*\*\*\* FILE SUMMARY  Input C:\Users\Adam Tran\Documents\CEE598 \_TrafficSimulation\gams\_codes\2\_S  hortest path problem.gms  Output C:\Users\Adam Tran\Documents\gamsdir\projdir\2\_Shortest path problem.  lst |

1. From Node 1 to Node 2:

|  |
| --- |
| VARIABLE x.L selection of flow between i and j  2 3  1 2.000  2  3 1.000  ---- VARIABLE z.L = 3.000 |

From Node 1 to Node 3:

|  |
| --- |
| VARIABLE x.L selection of flow between i and j  2 3  1 2.000  2  3  ---- VARIABLE z.L = 2.000 |

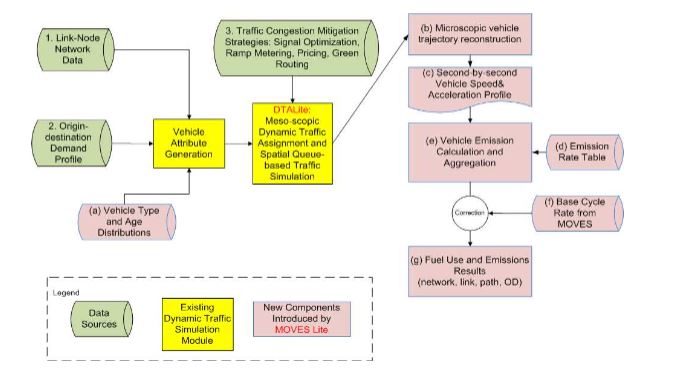
1. List the major input and output of dynamic traffic assignment, for traffic assignment and simulation steps. **(15 points)**, and clearly describe the corresponding optimization model (objective functions + all constraints) **(10 points)**

**Total: 25 points**.

Major inputs and outputs of Dynamic Traffic Assignment (DTA) for both traffic assignment and simulation steps:

|  |  |
| --- | --- |
| **Data or Progress** | **Corresponding Functions or Class** |
| Link-Node Network Data | Class: Node(), Link()  Functions: g\_read\_input\_data() |
| Origin-Destination Demand Profile | Class: Agent()  Functions: g\_read\_input\_data() |
| DTALite: Mesoscopic Dynamic Traffic Assignment  And Spatial Queue-Based Traffic Simulation | Class: Network()  Functions: g\_traffic\_assignment, g\_traffic\_simulation() |

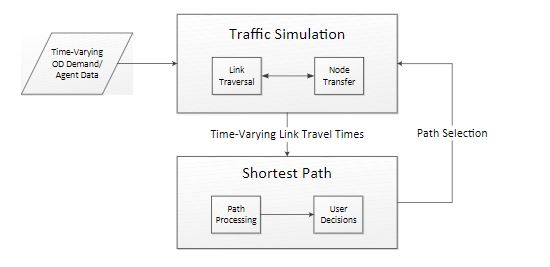
System Architecture and Data Flow:



Optimization model (objective function + all constraints):

DTA Modeling Framework

|  |  |
| --- | --- |
| **Item** | **Corresponding Functions or Steps** |
| Link Traversal | g\_traffic\_simulation() |
| Node Transfer | g\_traffic\_similation() |
| Path Processing | Network.optimal\_label\_correcting() |
| User Decision | Network.find\_path\_for\_agents() |



A Network Design Model for Accessibility:

* Space-time flow balance constraints
* Activity Constraints