BAMS 508 – Discrete Optimization

Assignment 2

Gurpal Bisra, Leon Zhu, Maja Peralta

Date: November 9, 2016

Question 1

1. Solving a TSP instance. Consider the instance of the Symmetric Traveling Salesman Problem (TSP) with twelve cities named A, B, ..., L and city-to-city distances given below. Choose one of the formulations given in class and estimate the number of variables and constraints if you were to write the full formulation explicitly. Solve this 12-city instance using the Relaxation & Separation (or Cutting Planes) method, starting with the degree constraints only. You may choose any strategy to solve each successive Relaxation problem and Separation problem, and you may use the computer to solve the relaxations. Detail and explain your calculations.

	A	B	C	D	E	F	G	H	I	J	K	L
A		24	26	20	23	21	25	18	22	17	29	20
B			23	28	10	17	11	27	7	25	26	17
C				30	11	10	24	29	21	16	17	8
D					27	25	29	22	26	22	32	24
E						17	10	26	8	12	25	16
F							19	24	16	22	12	2
G								28	8	26	27	18
H									25	21	32	23
I										23	24	15
J											30	21
K												11
L												

City-to-city distances (km)

Information Provided:

- n = 12 cities (i = 1, 2, … 12) to be visited which correspond to labels A, B, … L.
- distance c_{ij} to travel directly from city i to city j

We are asked to find a tour T starting at city 1, visiting each city exactly once and returning to city 1, with a minimum total length c(T) = $\sum_{ij\ \epsilon\ E(T)} c_{ij}$

Integer Programming Formulation:

Decision Variables:

- let $x_{ij} = \{1\ if\ the\ salesman\ travels\ directly\ from\ city\ i\ to\ j\ on\ tour\ T \atop 0\ otherwise\}$

Constraints:

1. **x binary**

2. **Tour:** The tour must visit every city once
   let $\delta(i)$ = { edges e : i is an endpoint of e }, the start of i
   $x(\delta(i)) = 2$ for all i ε V (STSP) Degree Constraints

Objective:

- To minimize the total length of the tour

min { $\sum_{i,j} c_{ij}x_{ij}$ } = Z

An annotated version of the Excel file is found below:

City-to-city distances:

	A	B	C	D	E	F	G	H	I	J	K	L
A		24	26	20	23	21	25	18	22	17	29	20
B			23	28	10	17	11	27	7	25	26	17
C				30	11	10	24	29	21	16	17	8
D					27	25	29	22	26	22	32	24
E						17	10	26	8	12	25	16
F							19	24	16	22	12	2
G								28	8	26	27	18
H									25	21	32	23
I										23	24	15
J											30	21
K												11
L												

Decision Variables

	A	B	C	D	E	F	G	H	I	J	K	L	Leaving Constraints			
A		0	0	1	0	0	0	1	0	0	0	0	1	2	=	2
B			0	0	0	0	1	0	1	0	0	0	2	2	=	2
C				0	1	0	0	0	0	1	0	0	3	2	=	2
D					0	0	0	1	0	0	0	0	4	2	=	2
E						0	0	0	0	1	0	0	5	2	=	2
F							0	0	0	0	1	1	6	2	=	2
G								0	1	0	0	0	7	2	=	2
H									0	0	0	0	8	2	=	2
I										0	0	0	9	2	=	2
J											0	0	10	2	=	2
K												1	11	2	=	2
L													12	2	=	2

Objective Function

min sum of cij*xij	150

We determined the following tour T exists:

- Subset S1: A→ D → H → A
- Subset S2: B → G → I → B
- Subset S3: C → J → E → C
- Subset S4: F → K → L → F

Next, we add additional cutset constraints according to:

STSP Cutset Constraints:

- for every nontrivial subset S of cities there must be at least one edge in the tour with one endpoint in S and the other in $V \setminus S$
  (C1) $x(E(S : V \setminus S)) \geq 1$ for every nontrivial subset S of cities
  where $E(S : V \setminus S)$ = {edges with exactly one endnode in S }
- identical to the cutset constraints for the spanning tree problem

3. **Clique Constraints**: the number of edges in the tour T with both endpoints in any nontrivial subset S of cities cannot exceed |S| - 1:

   x(E(S)) ≤ |S| - 1 for every nontrivial subset S of cities
   where E(S) = { edges in endpoints in S }

   C1: $x_{ah} + x_{dh} + x_{ad} \leq 2$
   C2: $x_{bi} + x_{bg} + x_{ig} \leq 2$
   C3: $x_{cj} + x_{ce} + x_{ej} \leq 2$
   C4: $x_{fl} + x_{fk} + x_{kl} \leq 2$

City-to-city distances:

	A	B	C	D	E	F	G	H	I	J	K	L
A		24	26	20	23	21	25	18	22	17	29	20
B			23	28	10	17	11	27	7	25	26	17
C				30	11	10	24	29	21	16	17	8
D					27	25	29	22	26	22	32	24
E						17	10	26	8	12	25	16
F							19	24	16	22	12	2
G								28	8	26	27	18
H									25	21	32	23
I										23	24	15
J											30	21
K												11
L												

Objective Function

min sum of cij*xij	153

Decision Variables

	A	B	C	D	E	F	G	H	I	J	K	L
A		0	0	0	0	0	0	1	0	1	0	0
B			0	0	1	0	0	0	1	0	0	0
C				0	0	0	0	0	0	0	1	1
D					0	0	0	1	0	1	0	0
E						0	1	0	0	0	0	0
F							0	0	0	0	1	1
G								0	1	0	0	0
H									0	0	0	0
I										0	0	0
J											0	0
K												0
L												

Clique Constraints

Leaving Constraints					
1	2	=	2		
2	2	=	2		
3	2	=	2		
4	2	=	2		
5	2	=	2		
6	2	=	2		
7	2	=	2		
8	2	=	2		
9	2	=	2		
10	2	=	2		
11	2	=	2		
12	2	=	2		
C1	9	2	≤	2	
C2	10	2	≤	2	
C3	11	0	≤	2	
C4	12	2	≤	2	

We determined the following tour T exists:

- Subset S1: A→ J → D → H → A
- Subset S2: B → I → G → E → B
- Subset S3: K → C → L → F → K

Next, we iterated new clique constraints and continued the problem using the following set of constraints:

4. **Clique Constraints**: the number of edges in the tour T with both endpoints in any nontrivial subset S of cities cannot exceed |S| - 1:

   x(E(S)) ≤ |S| - 1 for every nontrivial subset S of cities
   where E(S) = { edges in endpoints in S }

C5: $x_{ah} + x_{dh} + x_{dj} + x_{aj} + x_{ad} + x_{hj} \leq 3$
C6: $x_{bi} + x_{ig} + x_{ge} + x_{eb} + x_{be} + x_{gb} \leq 3$
C7: $x_{ck} + x_{kf} + x_{fl} + x_{cl} + x_{cf} + x_{kl} \leq 3$

City-to-city distances:

	A	B	C	D	E	F	G	H	I	J	K	L
A		24	26	20	23	21	25	18	22	17	29	20
B			23	28	10	17	11	27	7	25	26	17
C				30	11	10	24	29	21	16	17	8
D					27	25	29	22	26	22	32	24
E						17	10	26	8	12	25	16
F							19	24	16	22	12	2
G								28	8	26	27	18
H									25	21	32	23
I										23	24	15
J											30	21
K												11
L												

Objective Function

min sum of cij*xij	170

Decision Variables

	A	B	C	D	E	F	G	H	I	J	K	L
A		0	0	0	0	0	0	1	0	1	0	0
B			0	0	0	1	0	0	1	0	0	0
C				0	0	0	0	0	0	0	1	1
D					0	0	0	1	0	0	1	0
E						0	1	0	0	1	0	0
F							0	0	0	0	0	1
G								0	1	0	0	0
H									0	0	0	0
I										0	0	0
J											0	0
K												0
L												

Clique Constraints

Leaving Constraints					
1	2	=	2		
2	2	=	2		
3	2	=	2		
4	2	=	2		
5	2	=	2		
6	2	=	2		
7	2	=	2		
8	2	=	2		
9	2	=	2		
10	2	=	2		
11	2	=	2		
12	2	=	2		
C1	9	2	≤	2	
C2	10	2	≤	2	
C3	11	1	≤	2	
C4	12	1	≤	2	
C5	3	3	≤	3	
C6	10	3	≤	3	
C7	11	3	≤	3	

Hence, the solution is the following tour which has an objective value of 170 for distance:

A → H → D → K → C → L → F → B → I → G → E → J → A

Question 2

2. Portfolio partitioning. An investment dealer receives orders for an asset from his customers throughout the day. He also purchases units of the asset during the day at different prices. At the end of the day, the dealer wishes to allocate the assets to the customers in an "equitable way". Specifically, customer i (i = 1,...m) has placed orders for a total of d_i units of the asset today. The dealer has purchased the total amount $T = \Sigma_i\, d_i$ today in n lots, where lot j (j=1,...n) consists of s_j units at price p_j (dollars per unit). Therefore we have $\Sigma_j\, s_j = T$ and the total purchase cost $C = \Sigma_j\, p_j\, s_j$ is to be allocated "equitably" to the m customers. Ideally, the dealer would like to charge each customer i the average price C/T. However, current regulations require the dealer to allocate to each customer specific units of the asset at their purchase price.

One possible interpretation of "equitably" is to minimize the maximum, among all customers, of the resulting average price per unit charged to the customer. (As an illustrative example using the data below, if customer A gets 14 units from lot 1, 35 units from lot 3, and 1 unit from lot 4, she is charged a total of 14×995 + 35×1001 + 1×1006 = \$49,971 and thus an average price of 49,971/50 = \$999.42 per unit.)

(a) Show that this problem would have a trivial solution if fractional allocations of asset units were allowed. Find this "trivial solution" for the following instance.

Customer *i*	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Order *di* (units)	50	220	160	80	65	70	90	100	40	125

Lot j	**1**	**2**	**3**	**4**	**5**
Amount *sj* (units)	155	195	175	370	105
Purchase price *pj* (\$/unit)	995	989	1001	1006	999

In fact, due to the current regulations, each unit of the asset is indivisible and must be assigned to one customer. (For example, the dealer may *not* allocate 14.5 units from lot 1 to a customer.)

(b) Formulate a linear integer programming model for this problem. Show that the "trivial solution" found in (a) above solves the LP relaxation of this problem. Explain why this implies that this integer programming problem can be expected to be very difficult to solve using LP-based branch-and-bound.

(c) Using the *Excel Solver*, try and find a feasible integer solution with objective value within *half a cent* of the optimum. (What value of the percent Tolerance option should you use?) What happens?

(d) Repeat question (c) using the *OpenSolver*. What happens?

(e) Repeat question (c) using a mathematical programming system with an algebraic modeling language. Compare with your findings in (c) and (d) above.

(f) An alternate interpretation of "equitably" is to minimize the *sum*, over all T units, of the absolute deviations of the resulting prices per unit from the "target price" C/T. (In the earlier illustrative example, the absolute deviation from the target price was |999.42 – 999.37| = \$0.05 per unit for each of the 50 units allocated to customer A.) Note that summing over all units treats each unit of asset "equitably", and thus reflects the relative importance of the different customers, as indicated by their order quantities. Repeat questions (a) to (e) above for this new objective (for questions (c) to (e): within half a cent of the optimum on a per unit basis) and compare the results.

Problem 2

a) If fractional allocations of asset units were allowed, the asset units from each lot can simply be allocated to each customer based on the customers' share of ordered units in each lot, shown in the table below:

Decision Variables

Customer i / Lot j	A	B	C	D	E	F	G	H	I	J	Total	Amount s_j (units)	Purchase price p_j ($/unit)
1	7.75	34.1	24.8	12.4	10.075	10.85	13.95	15.5	6.2	19.375	155	155	995
2	9.75	42.9	31.2	15.6	12.675	13.65	17.55	19.5	7.8	24.375	195	195	989
3	8.75	38.5	28	14	11.375	12.25	15.75	17.5	7	21.875	175	175	1001
4	18.5	81.4	59.2	29.6	24.05	25.9	33.3	37	14.8	46.25	370	370	1006
5	5.25	23.1	16.8	8.4	6.825	7.35	9.45	10.5	4.2	13.125	105	105	999
Total	50	220	160	80	65	70	90	100	40	125			
Order d_i (units)	50	220	160	80	65	70	90	100	40	125		1000	999.37
Customer Share	0.05	0.22	0.16	0.08	0.065	0.07	0.09	0.1	0.04	0.125			

Objective Function

Average price ($/unit)	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37

Each decision variable value X_{ij} equals the available asset units in lot j multiplied by the ordering proportion for customer i. Such solution is a trivial one because no optimization is needed. Moreover, this method results to uniform average price per customer and therefore, minimizes their maximum value.

b)

Integer programming model formulation:

Decision Variables:
Let X_{ij} be the number of units of asset that are allocated to customer i, from lot j
Let Y be the maximum average price among i customers in $/unit

Objective function:
Minimize the maximum average price per customer among i customers

$$\text{Min } Y$$

(in $/unit)

Constraints:
1. The total units of assets from each lot for all customers should not exceed the available amount in each lot.

$$\sum_{i=A}^{J} X_{ij} \leq 155, \qquad for\ j = 1$$

$$\sum_{i=A}^{J} X_{ij} \leq 195, \qquad for\ j = 2$$

$$\sum_{i=A}^{J} X_{ij} \leq 175, \qquad for\ j = 3$$

$$\sum_{i=A}^{J} X_{ij} \leq 370, \qquad for\ j = 4$$

$$\sum_{i=A}^{J} X_{ij} \leq 105, \qquad for\ j = 5$$

2. The total units of assets for each customer, allocated from all lots, should not be less than the demanded amount by each customer.

$$\sum_{j=1}^{5} X_{ij} \geq 50, \qquad for\ i = A$$

$$\sum_{j=1}^{5} X_{ij} \geq 220, \qquad for\ i = B$$

$$\sum_{j=1}^{5} X_{ij} \geq 160, \qquad for\ i = C$$

...

$$\sum_{j=1}^{5} X_{ij} \geq 125, \qquad for\ i = J$$

3. The average price for each customer should not exceed the maximum average price Y.

$$\frac{\sum_{j=1}^{5}(X_{ij} * P_j)}{d_i} \leq Y, \qquad for\ i = A \ldots J$$

(in $/unit)

4. The number of units of asset X_{ij} for all i and j are integers.

5. The number of units of asset X_{ij} and maximum average price per customer Y are non-negative.

Solving for the LP Relaxation of this formulation will result to the same objective value (i.e. maximum average price at $ 999.37/unit) as the trivial solution. Excel Solver gives the optimal solution below:

Decision Variables

Customer i / Lot j	A	B	C	D	E	F	G	H	I	J	Total	Amount s_j (units)	Purchase price p_j ($/unit)
1	0	50.04091	29.61818	0	0	0	0	0	0	75.34091	155	155	995
2	19.5	18.47083	0	31.2	8.829167	27.3	35.1	39	15.6	0	195	195	989
3	0	118.8292	0	0	56.17083	0	0	0	0	0	175	175	1001
4	30.5	32.65909	25.38182	48.8	0	42.7	54.9	61	24.4	49.65909	370	370	1006
5	0	0	105	0	0	0	0	0	0	0	105	105	999
Total	50	220	160	80	65	70	90	100	40	125			
Order d_i (units)	50	220	160	80	65	70	90	100	40	125		1000	999.37

Objective Function

Average price ($/unit)	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	Objective
	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37

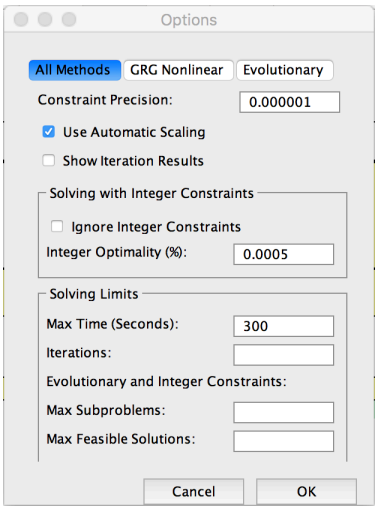
However, solving for the integer programming problem using this LP solution and branch-and-bound method will be too tedious to be done manually as 50 integer decision variables will mean having 2^{50} branches.

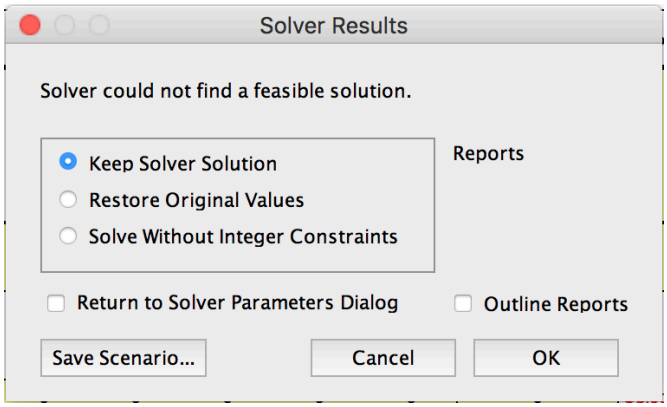
c)

The tolerance option in Excel Solver is the percentage of the true optimal solution that is allowed for the objective value to deviate from the optimum. Since objective value can be within half a cent of the optimum (which has the LP solution as its lower bound), the tolerance was obtained as follows:

$$\frac{0.005}{999.37} \times 100 = 0.0005\%$$

Using that tolerance value, Excel Solver could not find a feasible solution within 300 seconds.





d)

Using the same settings as in (e), OpenSolver did not find the true optimal solution as well. However, it did find a feasible integer solution within the 300-second limit.

Decision Variables

Customer i / Lot j	A	B	C	D	E	F	G	H	I	J	Total	Amount s_j (units)	Purchase price p_j ($/unit)
1	28	6	0	7	8	36	3	54	1	12	155	155	995
2	1	40	54	23	19	4	30	2	11	11	195	195	989
3	0	43	6	7	4	0	1	0	12	102	175	≤ 175	1001
4	20	60	84	39	34	30	49	39	15	0	370	370	1006
5	1	71	16	4	0	0	7	5	1	0	105	105	999
Total	50	220	160	80	65	70	90	100	40	125			
Order d_i (units)	≥ 50	220	160	80	65	70	90	100	40	125		1000	999.37

Objective Function

Average price ($/unit)	999.36	999.3727	999.375	999.3625	999.3692	999.3714	999.3667	999.37	999.375	999.368	999.375	Objective min
	≤ 999.375	999.375	999.375	999.375	999.375	999.375	999.375	999.375	999.375	999.375	999.375	999.375

e)

Setting the same tolerance level, in $ and not in % anymore, in AMPL, an optimal solution is obtained unlike in (c). However, it's objective value is a bit higher than that was obtained in (d) even after setting to MIP gap to $ 0.005.

```
CPLEX 12.6.3.0: absmipgap = 0.005
CPLEX 12.6.3.0: optimal integer solution within mipgap or absmipgap; objective 999.4615385
70 MIP simplex iterations
0 branch-and-bound nodes
absmipgap = 0.0915385, relmipgap = 9.15878e-05
No basis.
ampl: display Units;
Units [*,*]
:      1       2       3       4       5      :=
1    30.00    0.00    0.00   20.00    0.00
2     0.00   86.00    0.00  134.00    0.00
3    35.00   20.00   67.00   38.00    0.00
4     0.00   31.00    0.00   49.00    0.00
5     0.00   25.00    0.00   40.00    0.00
6     0.00    0.00    0.00    4.00   66.00
7    30.00    0.00    0.00   21.00   39.00
8    60.00    0.00    0.00   40.00    0.00
9     0.00   16.00    0.00   24.00    0.00
10    0.00   17.00  108.00    0.00    0.00
;

ampl: display Max_Avg_Price;
Max_Avg_Price = 999.46
```

f) Now the criteria for equity is to minimize the sum, over all T units, of the absolute deviations of the resulting price per unit from the "target price" C/T.

A)

Integer programming model formulation:

Decision Variables:

Let X_{ij} be the integer units of asset that are allocated to customer i, from lot j.

Let Y_i be the maximum between values of (Target price – Average price) and (Average price – Target price) for customer i, where $i \in A \dots J$.

Objective function:

Minimize the maximum average price among j customers.

Min $\sum_{i=A}^{J} Y_i$

Constraints:

1. The total units of assets from each lot for all customers should not exceed the available amount in each lot.

$$\sum_{i=A}^{J} X_{ij} \leq 155, \qquad for\ j = 1$$

$$\sum_{i=A}^{J} X_{ij} \leq 195, \qquad for\ j = 2$$

$$\sum_{i=A}^{J} X_{ij} \leq 175, \qquad for\ j = 3$$

$$\sum_{i=A}^{J} X_{ij} \leq 370, \qquad for\ j = 4$$

$$\sum_{i=A}^{J} X_{ij} \leq 105, \qquad for\ j = 5$$

2. The total units of assets for each customer, allocated from all lots, should not be less than the demanded amount by each customer.

$$\sum_{j=1}^{5} X_{ij} \geq 50, \qquad for\ i = A$$

$$\sum_{j=1}^{5} X_{ij} \geq 220, \qquad for\ i = B$$

$$\sum_{j=1}^{5} X_{ij} \geq 160, \qquad for\ i = C$$

...

$$\sum_{j=1}^{5} X_{ij} \geq 125, \qquad for\ i = J$$

3. The average price for each customer should not exceed the maximum average price Y.

$$\frac{\sum_{j=1}^{5}(X_{ij} * P_j)}{d_i} \leq Y_i, \qquad for\ i = A \ldots J$$

4. Integrality: all X_{ij} are integers

5. Non-negativity: all $X_{ij} \geq 0$

The trivial solution from previous part a) is still applicable here – it still solves the problem with minimum objective values, and all constraints satisfied.
Shown in the graph below:

Decision Variables

Customer *i*/ Lot *j*	A	B	C	D	E	F	G	H	I	J	Total	Amount s_j (units)	Purchase price p_j ($/unit)
1	7.75	34.1	24.8	12.4	10.075	10.85	13.95	15.5	6.2	19.375	155	155	995
2	9.75	42.9	31.2	15.6	12.675	13.65	17.55	19.5	7.8	24.375	195	195	989
3	8.75	38.5	28	14	11.375	12.25	15.75	17.5	7	21.875	175	175	1001
4	18.5	81.4	59.2	29.6	24.05	25.9	33.3	37	14.8	46.25	370	370	1006
5	5.25	23.1	16.8	8.4	6.825	7.35	9.45	10.5	4.2	13.125	105	105	999
Total	50	220	160	80	65	70	90	100	40	125			
Order d_i (units)	50	220	160	80	65	70	90	100	40	125		1000	999.37

Objective Function

target price 999.37

Average price ($/unit)	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	Objective
Average -Target	-4E-12	9.09495E-13	-1.3E-12	0	-1.0232E-12	-3.8654E-12	5E-12	0	0	1.36E-12		7.69718E-12
Target - Average	3.98E-12	-9.0949E-13	1.25E-12	0	1.02318E-12	3.86535E-12	-5E-12	0	0	-1.4E-12		
New DV	5.68E-14	7.63967E-12	6.66E-16	0	0	0	0	0	0	0		

B) The LP relaxation for this formulation solved is shown below:

Decision Variables

Customer i/ Lot j	A	B	C	D	E	F	G	H	I	J	Total	Amount s_j (units)	Purchase price p_j ($/unit)
1	0	25.29909091	96.43636	0	0	0	0	0	0	33.26455	155	155	995
2	6.791667	69.43	0	31.2	25.35	27.3	12.225	7.103333	15.6	0	195	195	989
3	43.20833	0	0	0	0	0	77.775	54.01667	0	0	175	175	1001
4	0	125.2709091	63.56364	48.8	39.65	42.7	0	0	24.4	25.61545	370	370	1006
5	0	0	0	0	0	0	0	38.88	0	66.12	105	105	999
Total	50	220	160	80	65	70	90	100	40	125			
Order d_i (units)	50	220	160	80	65	70	90	100	40	125		1000	999.37

Objective Function

target price
999.37

Average price ($/unit)	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	999.37	Objective
Average -Target	-4E-12	9.09495E-13	-1.3E-12	0	-1.0232E-12	-3.8654E-12	5E-12	0	0	1.36E-12		7.69718E-12
Target - Average	3.98E-12	-9.0949E-13	1.25E-12	0	1.02318E-12	3.86535E-12	-5E-12	0	0	-1.4E-12		
New constraints	5.68E-14	7.63967E-12	6.66E-16	0	0	0	0	0	0	0		

The decision variable values are presented in the table below.

	A	B	C	D	E	F	G	H	I	J
1	**0**	**25.299**	**96.436**	**0**	**0**	**0**	**0**	**0**	**0**	**33.265**
2	**6.791**	**69.43**	**0**	**31.2**	**25.35**	**27.3**	**12.225**	**7.103**	**15.6**	**0**
3	**43.20**	**0**	**0**	**0**	**0**	**0**	**77.775**	**54.017**	**0**	**0**
4	**0**	**125.271**	**63.563**	**48.8**	**39.65**	**42.7**	**0**	**0**	**24.4**	**25.615**
5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**38.88**	**0**	**66.12**

The trivial solution would still solve the LP relaxation because it results in the same average price value, with same objective function value of **7.69718E-12 (which is extremely close to 0)**, and all constraints satisfied.

Same reason as part b) previously, solving for the integer programming problem using this LP solution and branch-and-bound method will be too tedious to be done manually as 50 integer decision variables will mean having 2^{50} branches.

C) The integer program solution takes extremely long time to be solved with Excel Solver. If terminating the Solver half way during solving, we obtained following solution.

Decision Variables

Customer i/ Lot j	A	B	C	D	E	F	G	H	I	J	Total		Amount s_j (units)	Purchase price p_j ($/unit)
1	0	25	95	0	0	0	0	0	1	34	155		155	995
2	0	69	0	31	25	27	13	4	14	12	195		195	989
3	9	1	0	0	0	1	74	37	3	50	175	≤	175	1001
4	0	124	63	48	39	42	2	1	22	29	370		370	1006
5	41	1	2	1	1	0	1	58	0	0	105		105	999
Total	50	220	160	80	65	70	90	100	40	125				
Order d_i (units)	50	220	160	80	65	70	90	100	40	125			1000	999.37

Objective Function

	target price											
	999.37											
Average price ($/unit)	999.36	999.36364	999.381	999.33	999.35385	999.37143	999.36	999.41	999.4	999.38	**999.41**	Objective min 0.173211927
Average -Target	-0.01	-0.00636364	0.01125	-0.045	-0.01615385	0.001428571	-0.01444	0.04	0.03	0.006		
Target - Average	0.01	0.006363636	-0.01125	0.045	0.016153846	-0.00142857	0.014444	-0.04	-0.03	-0.006		
New constraints	0.01	0.006363636	0.01125	0.045	0.016153846	0	0.014444	0.04	0.03	0		

The integer decision variable values are as follows:

	A	B	C	D	E	F	G	H	I	J
1	**0**	**25**	**95**	**0**	**0**	**0**	**0**	**0**	**1**	**34**
2	**0**	**69**	**0**	**31**	**25**	**27**	**13**	**4**	**14**	**12**
3	**9**	**1**	**0**	**0**	**0**	**1**	**74**	**37**	**3**	**50**
4	**0**	**124**	**63**	**48**	**39**	**42**	**2**	**1**	**22**	**29**
5	**41**	**1**	**2**	**1**	**1**	**0**	**1**	**58**	**0**	**0**

These integer decision variable values yield an objective function value of 0.17321.

D) OpenSolver is not able to obtain a feasible solution for this formulation within a short computation time.

E) Implementing this algorithm in AMPL, we were not able to solve the model within 300 seconds. But the program is more efficient than Excel solvers and OpenSolvers and gave us better-optimized results.
A screenshot of AMPL solution is presented below:

```
ampl: include HW2_Pb2f.run
CPLEX 12.6.3.0:

<BREAK> (cplex)

<BREAK> (cplex)
aborted, integer solution exists; objective 0.03059931635
8173423 MIP simplex iterations
4532627 branch-and-bound nodes
absmipgap = 0.0256634, relmipgap = 0.838693
No basis.
amount [*,*]
:      1       2       3       4       5      :=
1     0.00    7.00   41.00    1.00    1.00
2    56.00   47.00    6.00  109.00    2.00
3    68.00   18.00    0.00   73.00    1.00
4     0.00   30.00    4.00   46.00    0.00
5     3.00   22.00    2.00   36.00    2.00
6     0.00   27.00    1.00   42.00    0.00
7     5.00   11.00   71.00    3.00    0.00
8    17.00   27.00    2.00   53.00    1.00
9     0.00    6.00   13.00    7.00   14.00
10    6.00    0.00   35.00    0.00   84.00
;
```

The decision variable values is organized in the following table:

The objective function value is **0.03059931635148415,** comparing to the objective function value from Excel Solver, which is 0.17321, this value from AMPL is more optimized.

	A	B	C	D	E	F	G	H	I	J
1	**0**	**56**	**68**	**0**	**3**	**0**	**5**	**17**	**0**	**6**
2	**7**	**47**	**18**	**30**	**22**	**27**	**11**	**27**	**6**	**0**
3	**41**	**6**	**0**	**4**	**2**	**1**	**71**	**2**	**13**	**35**
4	**1**	**109**	**73**	**46**	**36**	**42**	**3**	**53**	**7**	**0**
5	**1**	**2**	**1**	**0**	**2**	**0**	**0**	**1**	**14**	**84**

Question 3

3. Ranking Objects. We want to rank n objects (e.g., investment proposals, or sports teams) in a total order, from most to least preferred. We are given an $n \times n$ cost matrix C where the entry c_{ij} is the cost of ranking object i before j $(i \neq j)$. For example, c_{ij} is the number of managers who prefer investment j to i, or the number of games that team i has lost against team j. Thus we seek a total order of the n objects which minimizes the sum of the costs c_{ij} over all pairs (i, j) where i is ranked before j.

	A	B	C	D	E	F	G	H	I	J
A	.	4	2	0	1	2	2	3	0	0
B	0	.	4	2	0	4	4	2	2	1
C	2	0	.	4	1	2	1	2	0	2
D	4	2	0	.	1	4	3	4	2	1
E	3	4	3	3	.	1	4	2	1	0
F	2	0	2	0	3	.	2	2	4	0
G	2	0	3	1	0	2	.	1	4	2
H	1	2	2	0	2	2	3	.	1	1
I	4	2	4	2	3	0	0	3	.	2
J	4	3	2	3	4	4	2	3	2	.

(a) Formulate a binary integer programming model to determine a total order with minimum cost. Recall that a total order is irreflexive, antisymmetric and transitive. How many variables and constraints does your model contain? What are these model sizes for $n = 10$? (Using symmetry, can you reduce the number of transitivity constraints by a factor of 3? Explain.)

Information Provided:

- c_{ij} = the cost of ranking objects i before j (i ≠ j)
  - this matrix is asymmetric
- n = the number of objects, here n = 10 (i.e. A through J)
- the total order's relations must be:
  - Irreflexive meaning x_{ij} cannot have i = j for all i and j
  - Asymmetric meaning x_{ij} cannot have x_{ij} and x_{ji} both = 1 for all j and j
  - Transitive meaning for any *a*, *b*, and *c* in x_{ij}, whenever x_{ab} = 1 and x_{bc} = 1 then x_{ac} = 1
    http://www.cs.odu.edu/~toida/nerzic/content/relation/property/property.html

We are asked to formulate a binary integer programming model to determine the total order with minimum cost = $\sum_{ij} c_{ij}\, x_{ij}$

My model has 100 decision variables (i.e. n^2) and 875 constraints (i.e. $n + \frac{(n^2-n)}{2} + n^2 + n * (n-1) * (n-2)$) for n = 10 in our case.

Integer Programming Formulation:

Decision Variables: There will be a **total of 100 decision variables** for i = 1, 2, … 10 and j = 1, 2, … 10.

- let $x_{ij} = \begin{Bmatrix} 1\ if\ i\ is\ ordered\ before\ j \\ 0\ otherwise \end{Bmatrix}$

  where 1, 2, … 10 correspond to A, B, … J, respectively.
  (n^2 decision variables)

Constraints: a **total of 875 constraints are needed** to define the model as follows:

1. **10x Irreflexive Relation Constraints**: x_{ij} = 0 when i = j for all i and j **[constraints 1 – 10]**
   - $\boldsymbol{x_{ij} = 0\ for\ all\ i = j, i = 1, 2, \ldots n}$
     (n constraints)

2. **45x Asymmetric Relation Constraints**: x_{ij} cannot have x_{ij} and x_{ji} both = 1 for all j and j **[constraints 11 – 55]**
   - $\boldsymbol{x_{ij} + x_{ji} = 1\ for\ all\ i \neq j}$

     $\frac{(n^2-n)}{2}$ **constraints**

   - for example, x_{1j} adds 9 constraints:
     - x_{12} and $x_{21} \leq 1$
       …
     - $x_{1,10}$ and $x_{10,1} \leq 1$
   - x_{2j} adds 8 constraints
     …
   - x_{9j} adds 1 constraint
   - $x_{10,j}$ adds 0 constraints

3. **100x x_{ij} binary [constraints 56 - 155]**
   (n^2 decision variables)
   - All decision variables take on binary 1 or 0 values
   - x_{11} = 1 or 0
     …
   - $x_{10,10}$ = 1 or 0

4. **Transitive Relation**: for any *a*, *b*, and *c* in x_{ij}, whenever x_{ab} = 1 and x_{bc} = 1 then x_{ac} = 1 **[constraints 156 – 875]**
   - $\mathbf{0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \; for \; all \; i \neq j \neq k}$
     - formed by:

x_{ij}	x_{jk}	x_{ik}	Transitive Relation Satisfied?	$x_{ij} + x_{jk} - x_{ik}$
0	0	0	YES	0
1	0	0	YES	1
0	1	0	YES	1
0	0	1	NO	-1
0	1	1	YES	0
1	0	1	YES	0
1	1	0	NO	2
1	1	1	YES	1

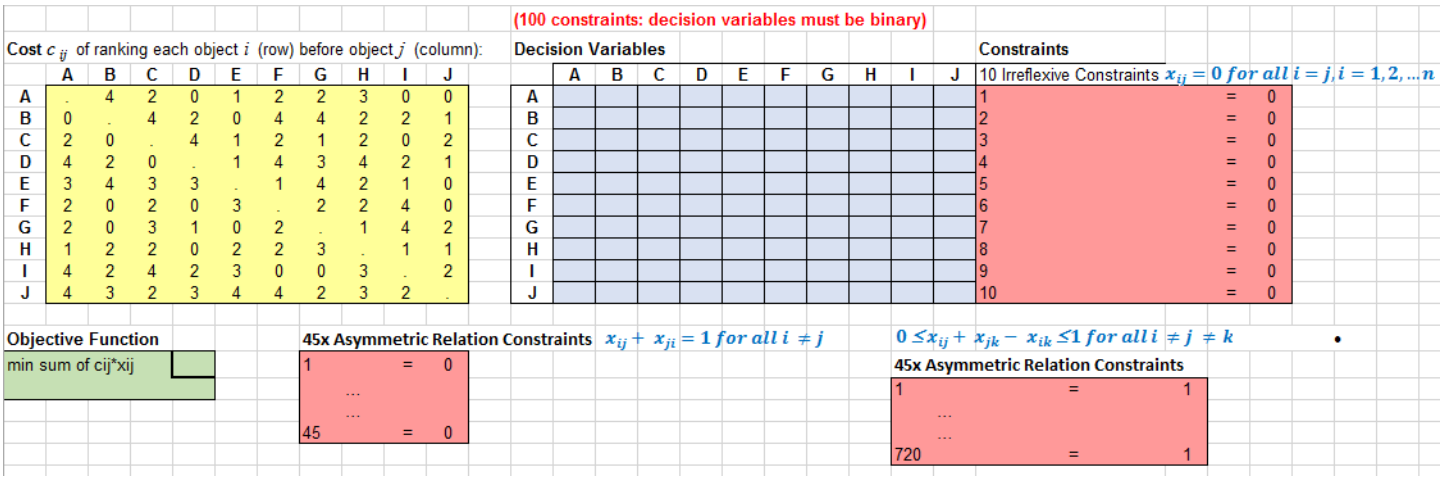
   - to count the number of constraints: $\boldsymbol{x_{ij} + x_{jk} - x_{ik}}$
     - i = 1, 2, … 10 so there are a total of 10 values
     - j = 1, 2, … 10 EXCEPT i ≠ j so there are a total of 9 values
     - k = 1, 2, … 10 EXCEPT k ≠ i ≠ j so there are a total of 8 values
   - **10*9*8 = 720 transitive relation constraints**
     ($\boldsymbol{n * (n-1) * (n-2)}$ constraints)
   - Each decision variable's value depends on all other values in the same row as well as their relations
     - Thought Process: 10x since suppose node 1 had the highest rank, then node 2, … and then node 10 (i.e. keep in mind that any of the labeled nodes are interchangeable), then row i = 1 can be filled as follows:
       - IF x_{12} = 1 and x_{23} = 1 THEN x_{13} = 1
       - IF x_{23} = 1 and x_{34} = 1 THEN x_{24} = 1
       - IF x_{34} = 1 and x_{45} = 1 THEN x_{35} = 1
       - IF x_{45} = 1 and x_{56} = 1 THEN x_{24} = 1
       - IF x_{56} = 1 and x_{67} = 1 THEN x_{24} = 1
       - IF x_{67} = 1 and x_{78} = 1 THEN x_{24} = 1
       - IF x_{78} = 1 and x_{89} = 1 THEN x_{24} = 1
       - IF x_{89} = 1 and $x_{9,10}$ = 1 THEN x_{24} = 1
       - IF $x_{9,10}$ = 1 and x_{12} = 1 THEN $x_{10,1}$ = 0
       - $\sum_j x_{ij} \leq 9$ [redundant constraint]
   - Using symmetry, **you cannot reduce the number of number of transivity constraints by 3** because one cannot predict where the ordering of rankings will start. For instance, if one knows x_{ab} = 1 and x_{bc} = 1 then x_{ac} = 1; however, this is always contingent upon the fact that x_{ab} = 1.

Objective:

- To minimize the total cost of the ordering of n = 10 factors.
  min { $\sum_{i,j} c_{ij} x_{ij}$ } = Z

(b) Without spending too much time on this question, try and define a spreadsheet model for your model of question (a). Ignoring any limitation on the number of variables, explain clearly, yet concisely, what main difficulty you would encounter with an instance size $n = 10$ or 20, and what you *could* do about it. You are *not* being asked to complete a spreadsheet implementation of your model.

For n = 10, I formulated the spreadsheet model as found below:



(100 constraints: decision variables must be binary)

Cost c_{ij} of ranking each object i (row) before object j (column):

	A	B	C	D	E	F	G	H	I	J
A	.	4	2	0	1	2	2	3	0	0
B	0	.	4	2	0	4	4	2	2	1
C	2	0	.	4	1	2	1	2	0	2
D	4	2	0	.	1	4	3	4	2	1
E	3	4	3	3	.	1	4	2	1	0
F	2	0	2	0	3	.	2	2	4	0
G	2	0	3	1	0	2	.	1	4	2
H	1	2	2	0	2	2	3	.	1	1
I	4	2	4	2	3	0	0	3	.	2
J	4	3	2	3	4	4	2	3	2	.

Decision Variables

	A	B	C	D	E	F	G	H	I	J
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

Constraints

10 Irreflexive Constraints $x_{ij} = 0 \; for \; all \; i = j, i = 1, 2, \ldots n$

1	=	0
2	=	0
3	=	0
4	=	0
5	=	0
6	=	0
7	=	0
8	=	0
9	=	0
10	=	0

Objective Function

min sum of cij*xij

45x Asymmetric Relation Constraints $x_{ij} + x_{ji} = 1 \; for \; all \; i \neq j$

1	=	0
...		
45	=	0

$0 \leq x_{ij} + x_{jk} - x_{ik} \leq 1 \; for \; all \; i \neq j \neq k$

45x Asymmetric Relation Constraints

1	=	1
...		
720	=	1

For n = 20, it is very difficult to formulate the spreadsheet model because one would need to generate:

- $\boldsymbol{n + \frac{(n^2 - n)}{2} + n^2 + n * (n - 1) * (n - 2) = 20 + \frac{(20^2 - 20)}{2} + 20^2 + 20 * (20 - 1) * (20 - 2) =}$ **7450 constraints**
- **Problems**:
  1. The main problem is that when the number of nodes increases, the number of transitivity constraints grows as fast as n*(n-1)*(n-2). Such a high number of transitivity constraints would take tremendous time to write in an Excel spreadsheet; the odds of making a typo are high too.
  2. Excel Solver might not be able to handle so many constraints simultaneously (i.e. not finish executing)
- **Solutions:**
  1. One could write a VBA Macro in Excel to fill in the spreadsheet with all of the constraints
  2. One could modify the Excel OpenSolver macro itself and incorporate Warshall's Algorithm to fill the Transitivity Constraint (only if this is faster than OpenSolver's current algorithm). Then, the Transitivity Constraint can check which combination works.

ALGORITHM *Warshall*$(A[1..n, 1..n])$

//Implements Warshall's algorithm for computing the transitive closure
//Input: The adjacency matrix A of a digraph with n vertices
//Output: The transitive closure of the digraph
$R^{(0)} \leftarrow A$
for $k \leftarrow 1$ **to** n **do**
    for $i \leftarrow 1$ **to** n **do**
        for $j \leftarrow 1$ **to** n **do**
            $R^{(k)}[i, j] \leftarrow R^{(k-1)}[i, j]$ **or** $(R^{(k-1)}[i, k]$ **and** $R^{(k-1)}[k, j])$
return $R^{(n)}$

(c) Formulate your integer programming model using an algebraic modeling language. Make sure to use such language constructs as indexed variables and constraints; summations; and data tables or data files. Briefly comment on the relative ease of implementation of a model of this type using an algebraic modeling language compared with a spreadsheet implementation.

*order_model.mod | order_data.dat | order_run.run

```
# RUN FILE FOR QUESTION 3

reset;      # clears AMPL memory, to allow repeated runs
            # within the same AMPL session
model order_model.mod;
data  order_data.dat;

option display_1col 0;  # for compact display
option display_round 2; # output formatting (rounded to 2 decimal digits)

# Next, solve the IP model using the CPlex MIP solver:
option solver cplex;
solve;

display x;
    # (Note: the LP shadow prices are meaningless when solving an IP)

# include order_run.run
```

*order_model.mod | order_data.dat | order_run.run

```
# data file for the question 3

set NODE := 'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J';

param COSTMATRIX:
        'A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' :=
'A'     0   4   2   0   1   2   2   3   0   0
'B'     0   0   4   2   0   4   4   2   2   1
'C'     2   0   0   4   1   2   1   2   0   2
'D'     4   2   0   0   1   4   3   4   2   1
'E'     3   4   3   3   0   1   4   2   1   0
'F'     2   0   2   0   3   0   2   2   4   0
'G'     2   0   3   1   0   2   0   1   4   2
'H'     1   2   2   0   2   2   3   0   1   1
'I'     4   2   4   2   3   0   0   3   0   2
'J'     4   3   2   3   4   4   2   3   2   0;
```

*order_model.mod | order_data.dat | order_run.run

```
# DATA

set NODE;
param COSTMATRIX {NODE, NODE} >= 0;

# DECISION VARIABLES
var x{NODE, NODE} >= 0;     # Integer Decision Variables
# var x{NODE, NODE} >= 0;   # Binary Decision Variables

# OBJECTIVE FUNCTION
minimize total_order_cost : sum{ i in NODE, j in NODE} COSTMATRIX[i,j]*x[i,j] ;

# CONSTRAINTS
# 10x Irreflexive Relation Constraints:
# xij = 0 when i = j for all i and j
subject to Irreflexive {i in NODE}:
x[i,i]=0;

# 45x Asymmetric Relation Constraints:
# xij cannot have xij and xji both = 1 for all j and j
subject to Asymmetric {i in NODE, j in NODE diff {i}}:  # set difference to exclude i
x[i,j]+x[j,i]=1;

# 7 Transitive Relation:
# for any a, b, and c in xij, whenever xab = 1 and xbc = 1 then xac = 1
subject to Transitive_0value {i in NODE, j in (NODE diff {i}), k in (NODE diff {i,j})}:
(x[i,j]+x[j,k]-x[i,k])>=0;

subject to Transitive_1value {i in NODE, j in (NODE diff {i}), k in (NODE diff {i,j})}:
(x[i,j]+x[j,k]-x[i,k])<=1;
```

Using AMPL instead of Excel OpenSolver was much easier because the transitive relation constraints can be written in 4 lines instead of programming 720 individual cells. Likewise, programming the other two constraints (i.e. Reflexive Relation and Asymmetric Relation constraints) requires coding 4 lines instead of filling out 55 individual Excel files. On the other hand, setting the decision variables to binary is not any

different in terms of ease. Overall, implementing IF/ELSE statements and FOR loops appears much easier in AMPL as compared to Excel which would require, at minimum, VBA. For scaling the number of nodes, the Excel Spreadsheet would require tremendous effort, be difficult to debug, and significantly more time consuming to construct. On the other hand, scaling the AMPL program to contain more ranks requires minimal changes such as expanding the data file to include more data.

(d) Using your algebraic model of question (c) and a corresponding mathematical programming system, solve both the LP relaxation and the integer program for the 10-object instance with cost matrix shown below. (For example, the cost of ranking object **A** before **B** is 4. This cost matrix is also available on the course web site.) Comment on the differences (if any) between the LP and integer solutions. Make sure to produce and explain all relevant computer printouts.

The solution for the LP relaxation (i.e. setting x_{ij} to integer values instead of binary) is as follows:

```
ampl: include order_run.run
CPLEX 12.6.3.0: optimal solution; objective 56
143 dual simplex iterations (82 in phase I) on the dual problem
x [*,*]
:    A      B      C      D      E      F      G      H      I      J      :=
A   0.00   0.50   0.50   1.00   1.00   0.50   0.50   0.00   1.00   1.00
B   0.50   0.00   0.00   0.50   1.00   0.00   0.00   0.00   0.50   1.00
C   0.50   1.00   0.00   0.50   1.00   0.50   0.50   0.00   1.00   1.00
D   0.00   0.50   0.50   0.00   1.00   0.00   0.00   0.00   0.50   1.00
E   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.50   1.00
F   0.50   1.00   0.50   1.00   1.00   0.00   0.50   0.00   0.50   1.00
G   0.50   1.00   0.50   1.00   1.00   0.50   0.00   0.50   0.50   1.00
H   1.00   1.00   1.00   1.00   1.00   1.00   0.50   0.00   1.00   1.00
I   0.00   0.50   0.00   0.50   0.50   0.50   0.50   0.00   0.00   1.00
J   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
;
```

In contrast, the solution for the binary programming model (i.e. setting x_{ij} to binary values instead of integer) is as follows:

```
ampl: include order_run.run
CPLEX 12.6.3.0: optimal integer solution; objective 57
34 MIP simplex iterations
0 branch-and-bound nodes
No basis.
x [*,*]
:    A      B      C      D      E      F      G      H      I      J      :=
A   0.00   1.00   1.00   1.00   1.00   1.00   1.00   0.00   1.00   1.00
B   0.00   0.00   0.00   1.00   1.00   0.00   0.00   0.00   0.00   1.00
C   0.00   1.00   0.00   1.00   1.00   1.00   1.00   0.00   1.00   1.00
D   0.00   0.00   0.00   0.00   1.00   0.00   0.00   0.00   0.00   1.00
E   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   1.00
F   0.00   1.00   0.00   1.00   1.00   0.00   1.00   0.00   0.00   1.00
G   0.00   1.00   0.00   1.00   1.00   0.00   0.00   0.00   0.00   1.00
H   1.00   1.00   1.00   1.00   1.00   1.00   1.00   0.00   1.00   1.00
I   0.00   1.00   0.00   1.00   1.00   1.00   1.00   0.00   0.00   1.00
J   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00   0.00
;
```

The only difference I see is that the values of x_{ij} round up from 0.5 to 1 while the corresponding values of x_{ji} round down to 0. This satisfies the asymmetry constraint. In addition, the objective value for the LP relaxation is 56 whereas the objective value is 57 for the binary programming model.

One can interpret the results of a rank (i.e. for the "nodes" A through J) by summing the number of binary 1's in each row to determine it's rank. For instance:

A	8
B	3
C	7
D	2
E	1
F	5
G	4
H	9
I	6
J	0

Hence, the highest to lowest rank, at a cost of 57, is as follows:
H > A > C > I > F > G > B > D > E > J