ISyE6669, Fall 2016 - Team Project 2 Report

D'Souza, Ajay
ajaydsouza@gatech.edu

Huestis, Sarah
shuestis3@gatech.edu

Li, Albert
albertli@gatech.edu

Yllander, Sean
syllander3@gmail.com

Project 2 Report

Abstract

Project 2 Report for ISyE6669 Deterministic Optimization

Contents

List of Figures

List of Tables

1 Computation Experiments - Q1.4

1.1 No Small Sub-tour Elimination Constraints - Q1.4.1

The 48 city TSP tour with commenting out all the small Sub-tour constraints completed its run in 36.91 seconds. The number of generated Sub-Tour constraints during the course of the run was 171. The TSP had a optimal distance of 46837.9. The TSP Tour for this formulation is plotted in figure (1) below. The results of this formulation are tabulated in table (1)

1.2 1-City Sub-tour Elimination Constraint- Q1.4.2

The 48 city TSP tour initialized with only the 1-City small sub-tour elimination constraint, completed its run in 31.68 seconds. The number of generated Sub-Tour constraints during the course of the run was 146. The TSP had a optimal distance of 50521.3 . The TSP Tour for this formulation is plotted in figure (2) below. The results of this formulation are tabulated in table (1)

1.3 2-City Sub-tour Elimination Constraint - Q1.4.3

The 48 city TSP tour initialized with both the 1-City and 2-City small sub-tour elimination constraints, completed its run in 30.75 seconds. The number of generated Sub-Tour constraints during the course of the run was 51. The TSP had a optimal distance of 39829.1 . The TSP Tour for this formulation is plotted in figure (3) below. The results of this formulation are tabulated in table (1)

1.4 3-City Sub-tour Elimination Constraint - Q1.4.4

The 48 city TSP tour initialized with the 1-City 2-City and 3-City small sub-tour elimination constraints, completed its run in 119.35 seconds. The number of generated Sub-Tour constraints during the course of the run was 27. The TSP had a optimal distance of

35207.9 . The TSP Tour for this formulation is plotted in figure (4) below. The results of this formulation are tabulated in table (1)

1.5 4-City Sub-tour Elimination Constraint - Q1.4.5

The 48 city TSP tour initialized with the 1-City 2-City, 3-City and 4-City small sub-tour elimination constraints, completed its run in 3840.46 seconds ($\approxeq 64Mins$). The number of generated Sub-Tour constraints during the course of the run was 33. The TSP had a optimal distance of 35868.7 . The TSP Tour for this formulation is plotted in figure (5) below. The results of this formulation are tabulated in table (1)

1.6 Comparing the Results for each formulation for the 48 City Tour - Q1.4.6

Table (1) tabulates the results of the time taken in seconds, the number of constraints generated and the optimal tour distance for the 48 city tour for each of the formulations with different initial sub-tour constraints

Formulation	Time Taken(Secs)	No. of Generated Sub-tour Constraints	Optimal Tour Distance
No Sub-tour Constraint	36.91	171	46837.9
1 City Sub-tour Constraint	31.67	146	50521.3
2 City Sub-tour Constraint	30.75	51	39829.1
3 City Sub-tour Constraint	119.35	27	35207.9
4 City Sub-tour Constraint	3840.46	33	35868.7

Table 1: 48 City Tour - Compare results from formulations with different Initial Sub-tour Constraints

1.7 48 City TSP Tour plots - Q1.4.7

The following are the plots of the tour generated by the various formulations for the 48 City tour.

Figure (1) is the plot of the TSP of the 48 city tour with no initial Sub-tour constraints.

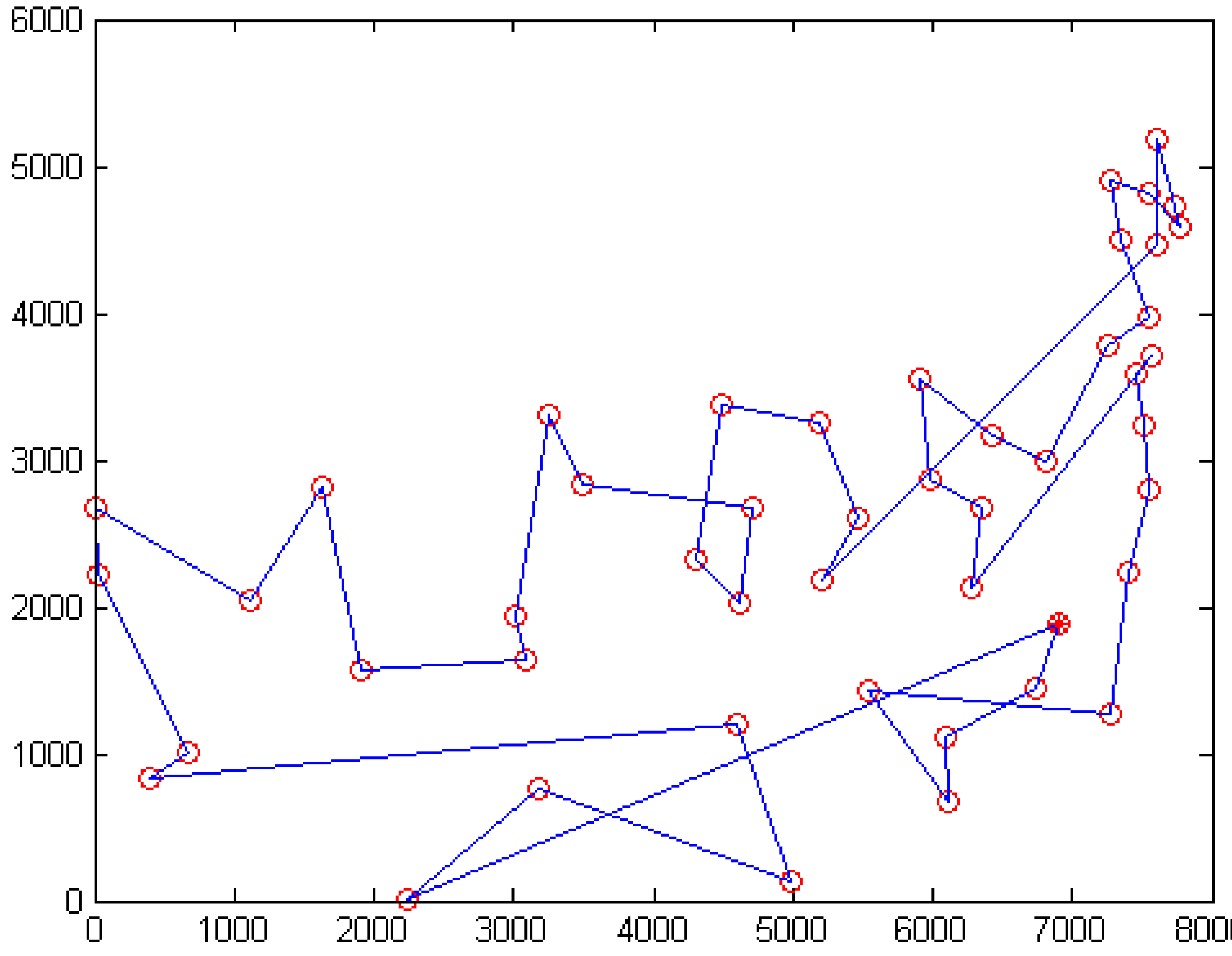

Figure 1: TSP Tour Plot - 48 City Tour with no initial Sub-tour constraints

Figure (2) is the plot of the TSP of the 48 city tour with 1-City Sub-tour constraint.

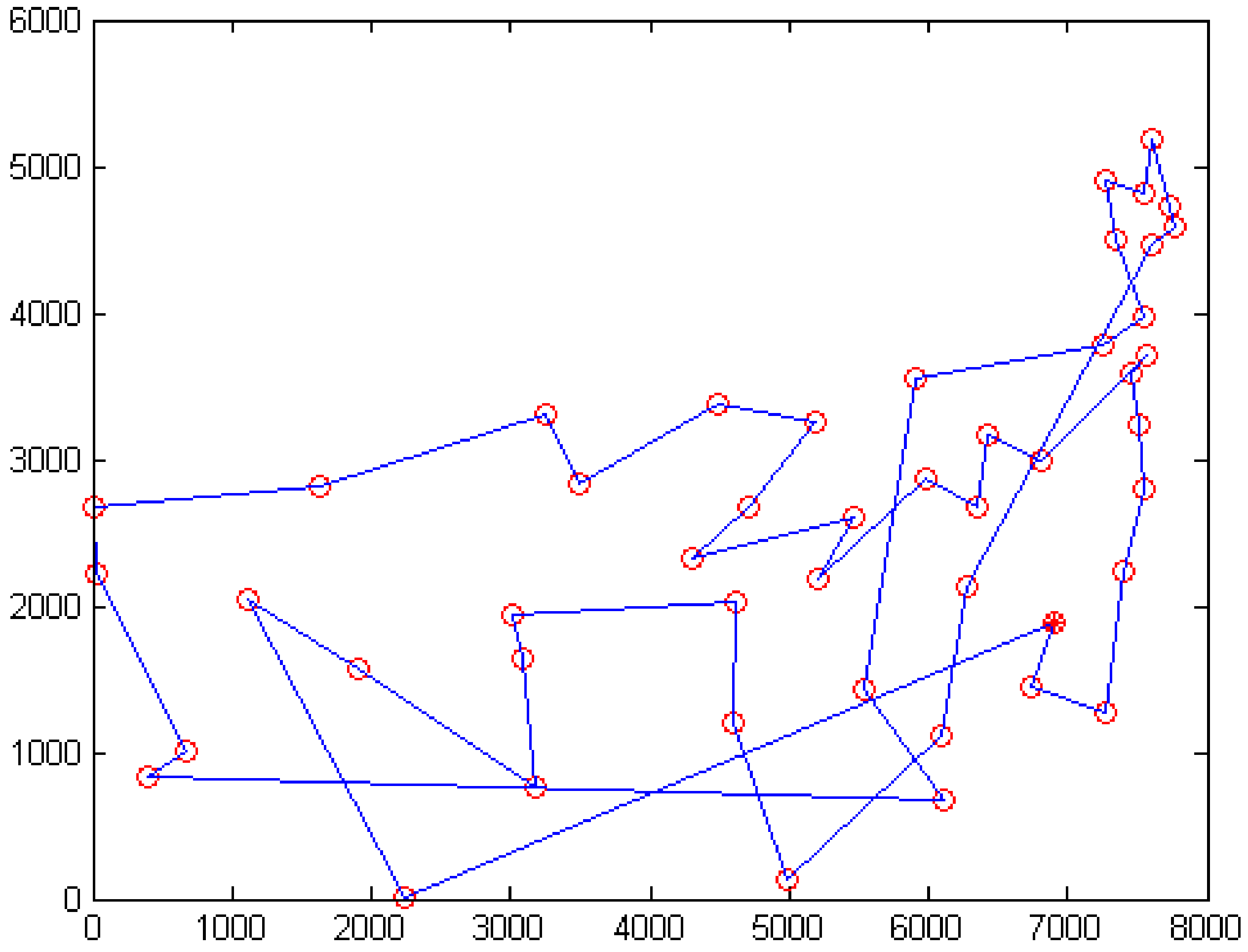

Figure 2: TSP Tour Plot - 48 City Tour with 1-City initial Sub-tour constraints

Figure (3) is the plot of the TSP of the 48 city tour with 2-City Sub-tour constraint.

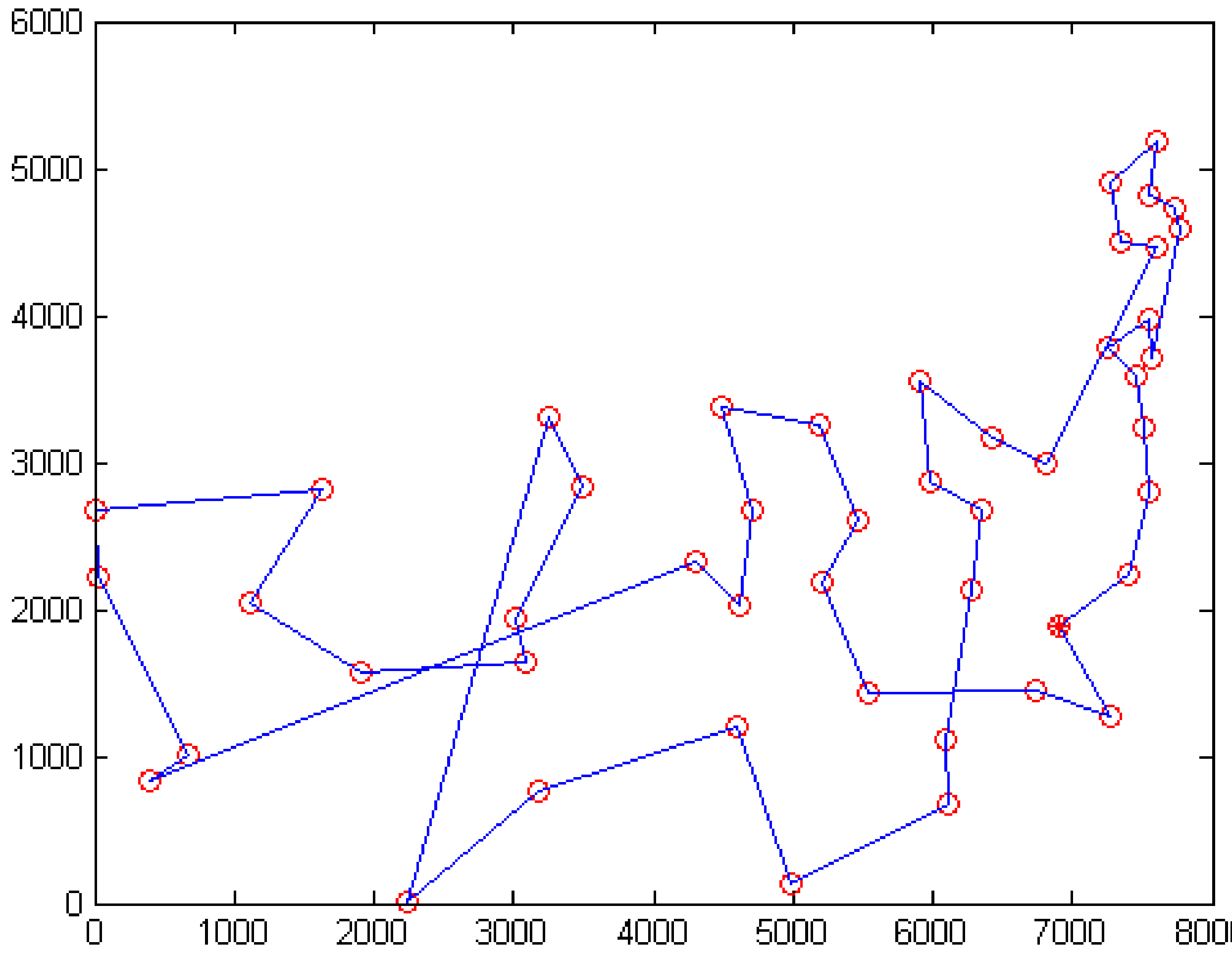

Figure 3: TSP Tour Plot - 48 City Tour with 2-City initial Sub-tour constraints

Figure (4) is the plot of the TSP of the 48 city tour with 3-City Sub-tour constraint.

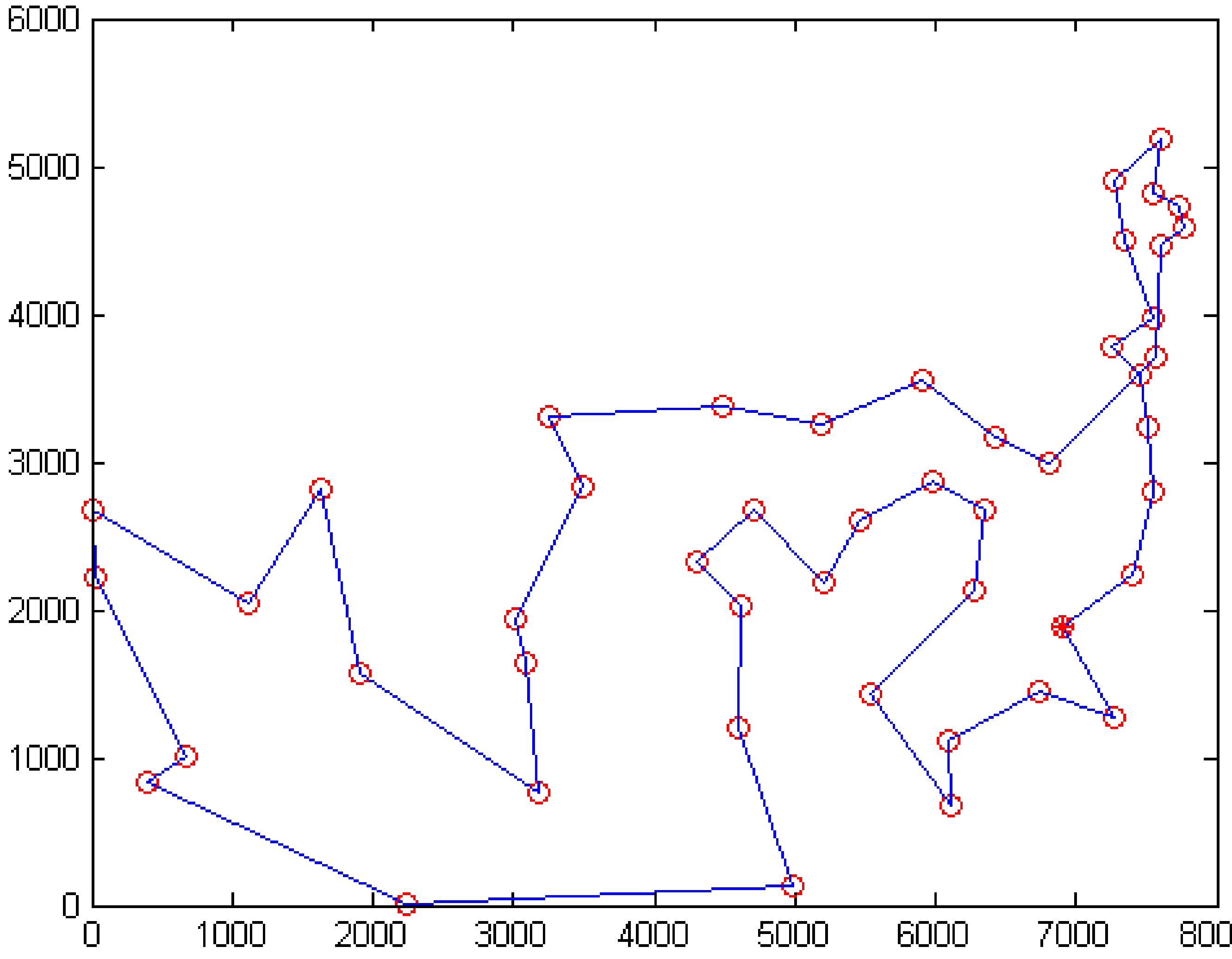

Figure 4: TSP Tour Plot - 48 City Tour with 3-City initial Sub-tour constraints

Figure (5) is the plot of the TSP of the 48 City tour with 4-City Sub-tour constraint.

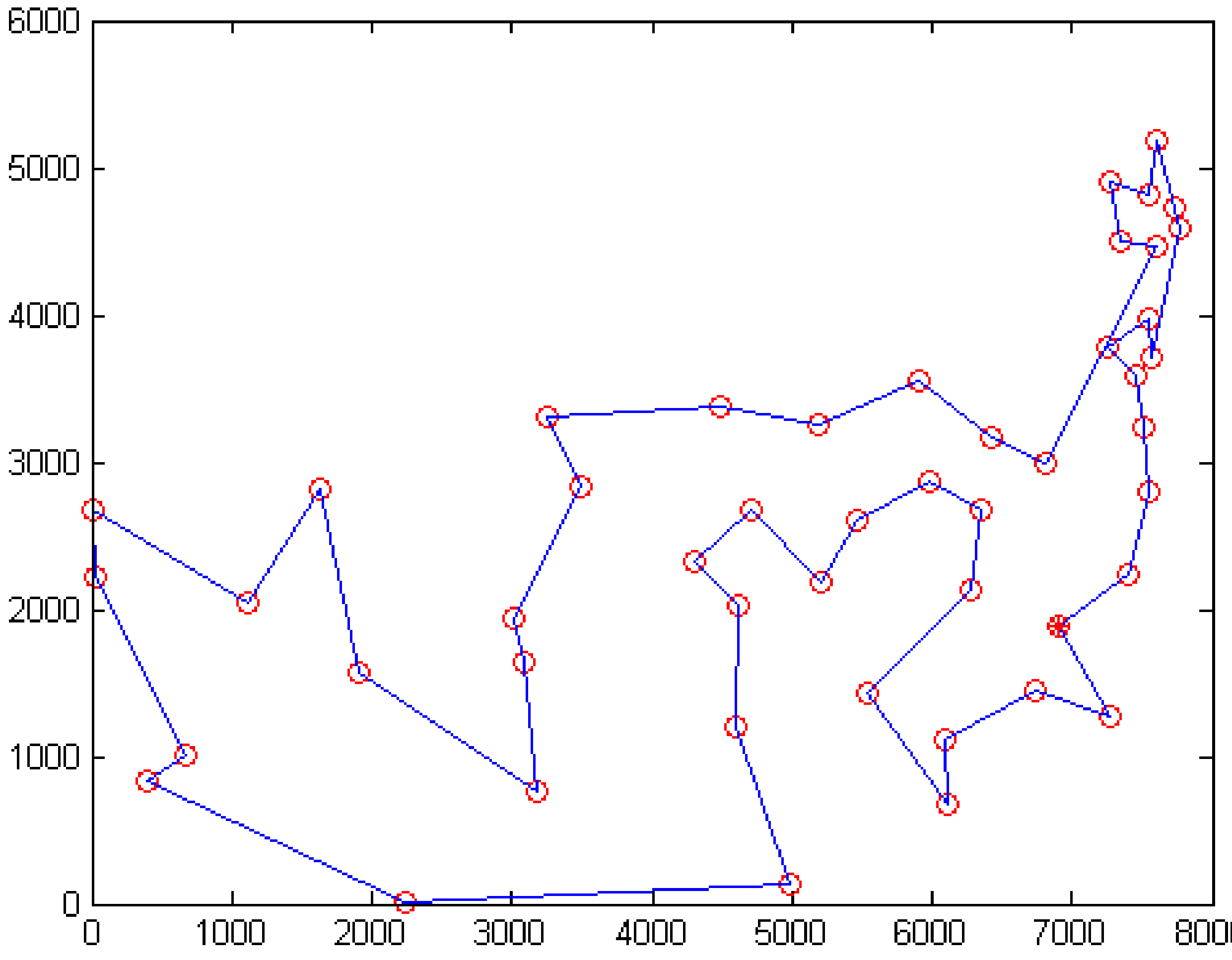

Figure 5: TSP Tour Plot - 48 City Tour with 4-City initial Sub-tour constraints

1.8 48 City TSP Tour with City Names - Q1.4.8

The following table (2) lists the city names used for the TSP tour.

No	City Name		No	City Name
1	Atlanta		25	Lincoln
2	Phoenix		26	Carson City
3	Little Rock		27	Concord
4	Sacramento		28	Trenton
5	Denver		29	Santa Fe
6	Hartford		30	Albany
7	Dover		31	Raleigh
8	Tallahassee		32	Bismark
9	Montgomery		33	Columbus
10	Boise		34	Oklahoma City
11	Springfield		35	Salem
12	Indianapolis		36	Harrisburg
13	Des Moines		37	Providence
14	Topeka		38	Columbia
15	Frankfort		39	Pierre
16	Baton Rouge		40	Nashville
17	Augusta		41	Austin
18	Annapolis		42	Salt Lake City
19	Boston		43	Montpelier
20	Lansing		44	Richmond
21	Saint Paul		45	Olympia
22	Jackson		46	Charleston
23	Jefferson City		47	Madison
24	Helena		48	Cheyenne

Table 2: 48 City Tour - List of Cities

The formulation with 3-City small sub-tour initial constraints gives optimum tour with the lowest distance of 35207.9, for the 48 City TSP tour. Figure (6) is the plot of the TSP of the best 48 City tour with city names obtained with 3-City small initial Sub-tour constraint.

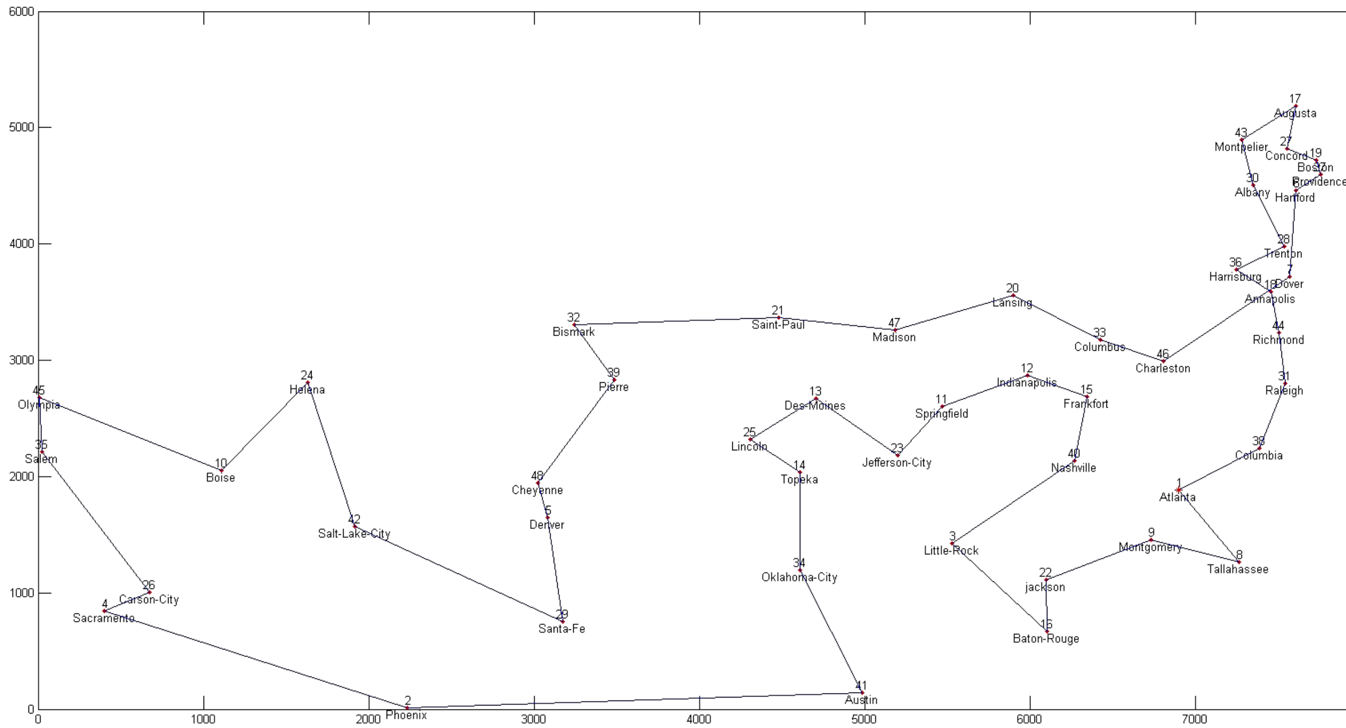

Figure 6: Best TSP Tour Plot - 48 City Tour with 3-City initial Sub-tour constraints

1.9 24 City TSP Tour - Q1.4.9

The following are the 24 cities randomly chosen for the 24 city tour

```
coord  : [
1 6898 1885
3 5530 1424
4 401 841
5 3082 1644
6 7608 4458
8 7265 1268
11 5468 2606
15 6347 2683
16 6107 669
19 7732 4723
20 5900 3561
21 4483 3369
23 5199 2182
26 675 1006
30 7352 4506
31 7545 2801
34 4608 1198
35 23 2216
38 7392 2244
40 6271 2135
43 7280 4899
44 7509 3239
47 5185 3258
48 3023 1942]
```

1.9.1 Comparing the Results for each formulation for the 24 City Tour

The 24 City tour was done using the different initial small sub-tour constraints from no city to 4-City.

Table (3) tabulates the results of the time taken in seconds, the number of constraints generated and the optimal tour distance for the 24 city tour for each of the formulations with different initial sub-tour constraints

Formulation	Time Taken(Secs)	No. of Generated Sub-tour Constraints	Optimal Tour Distance
No Sub-tour Constraint	4.56	72	33974.5
1 City Sub-tour Constraint	1.25	17	26876.8
2 City Sub-tour Constraint	1.54	12	25695.4
3 City Sub-tour Constraint	5.43	13	28413.2
4 City Sub-tour Constraint	33.35	5	25353.6

Table 3: 24 City Tour - Compare results from formulations with the different Initial Sub-tour Constraints

1.9.2 24 City Tour - Plotting the Results for each formulation

The following figures plot the 24 City tours generated with each different small sub-tour constraints. The formulation with 4-City small sub-tour initial constraints gives optimum tour with the lowest distance of 25353.6, for the 24 City TSP tour.

Figure (7) is the plot of the TSP of the 24 city tour with no initial Sub-tour constraints.

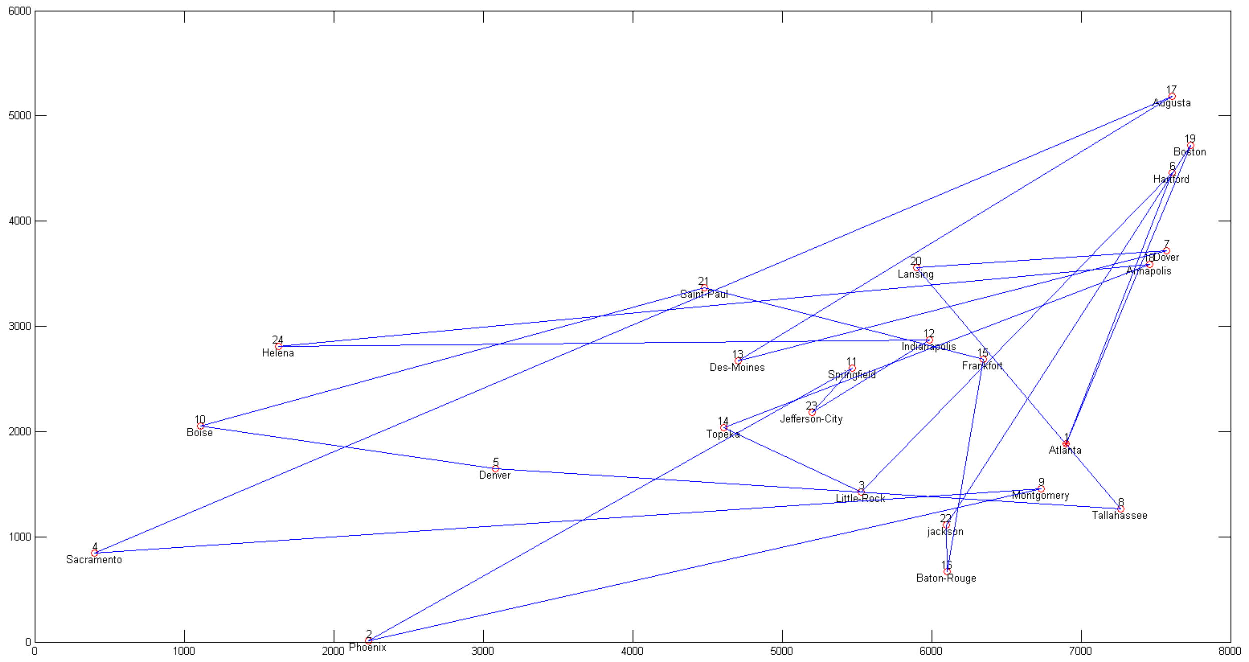

Figure 7: TSP Tour Plot - 24 City Tour with no initial Sub-tour constraints

Figure (8) is the plot of the TSP of the 24 city tour with 1-City Sub-tour constraint.

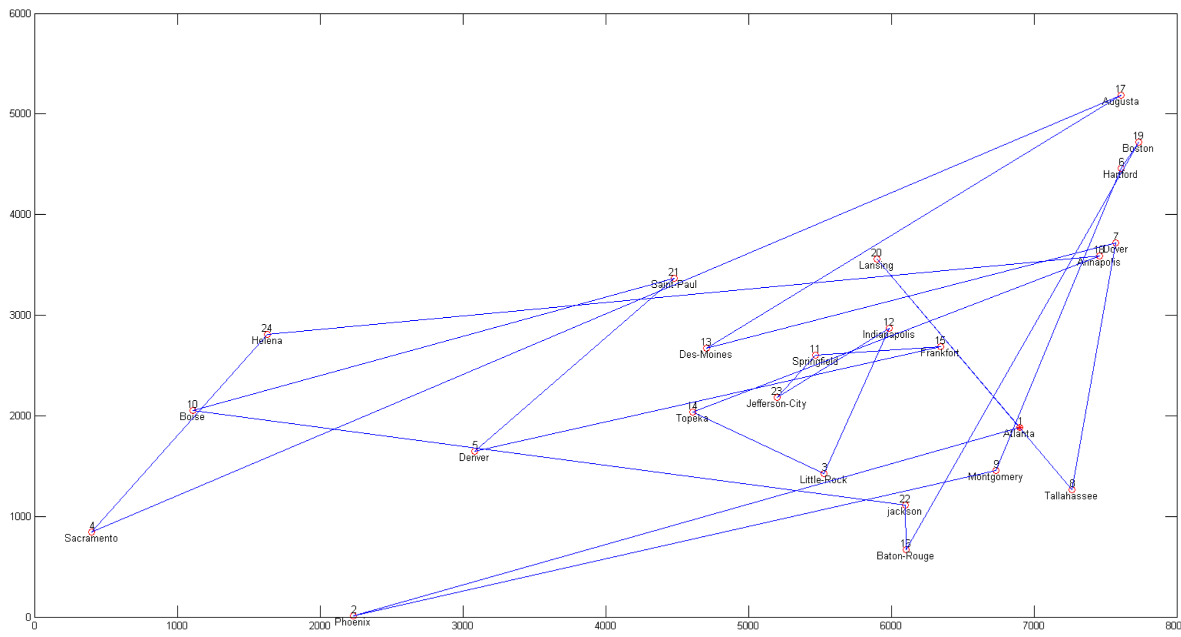

Figure 8: TSP Tour Plot - 24 City Tour with 1-City initial Sub-tour constraints

Figure (9) is the plot of the TSP of the 24 city tour with 2-City Sub-tour constraint.

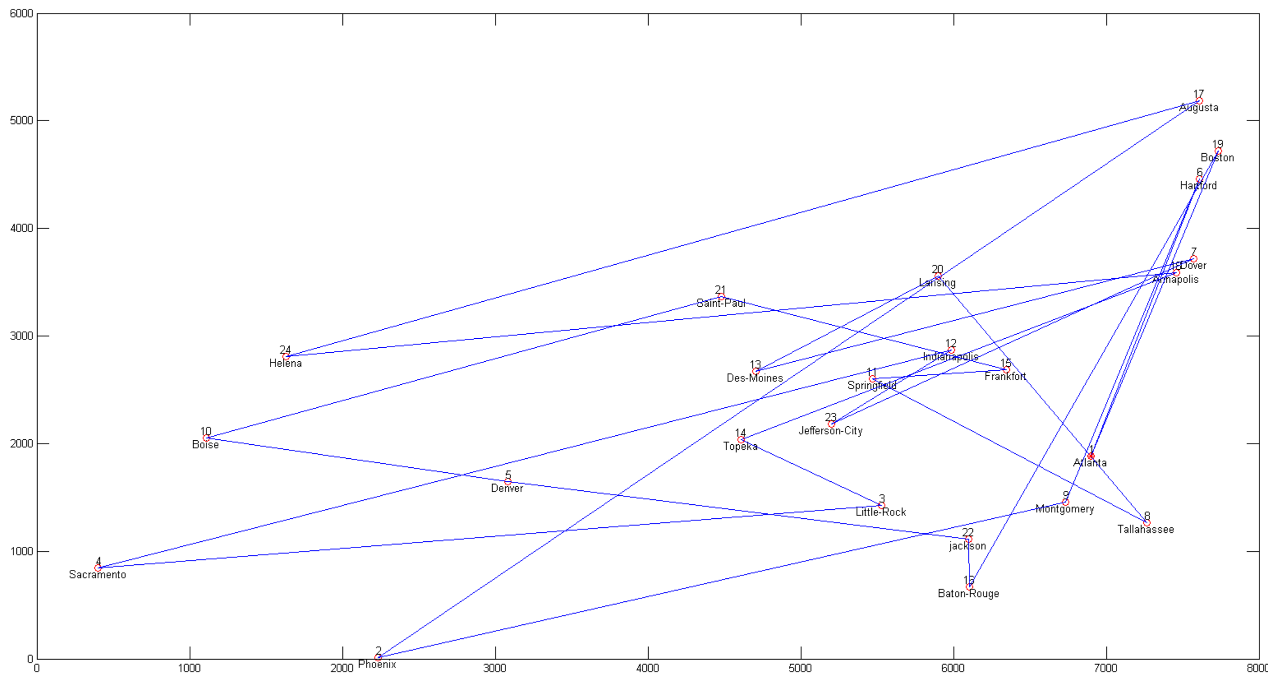

Figure 9: TSP Tour Plot - 24 City Tour with 2-City initial Sub-tour constraints

Figure (10) is the plot of the TSP of the 24 city tour with 3-City Sub-tour constraint.

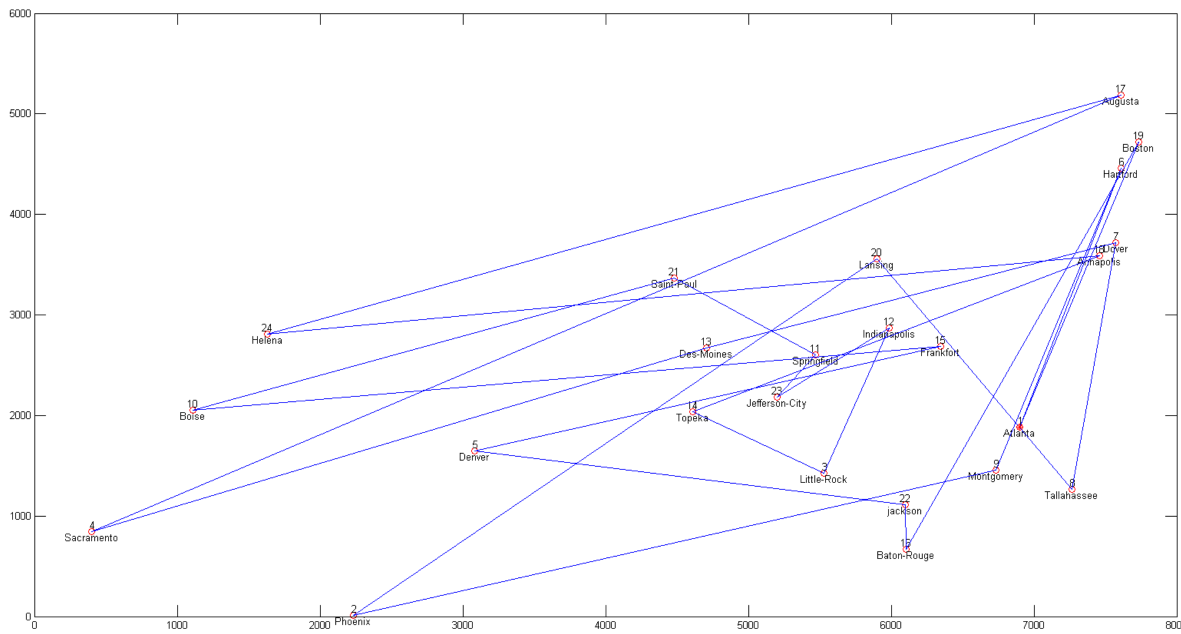

Figure 10: TSP Tour Plot - 24 City Tour with 3-City initial Sub-tour constraints

Figure (11) is the plot of the TSP of the 24 city tour with 4-City Sub-tour constraint.

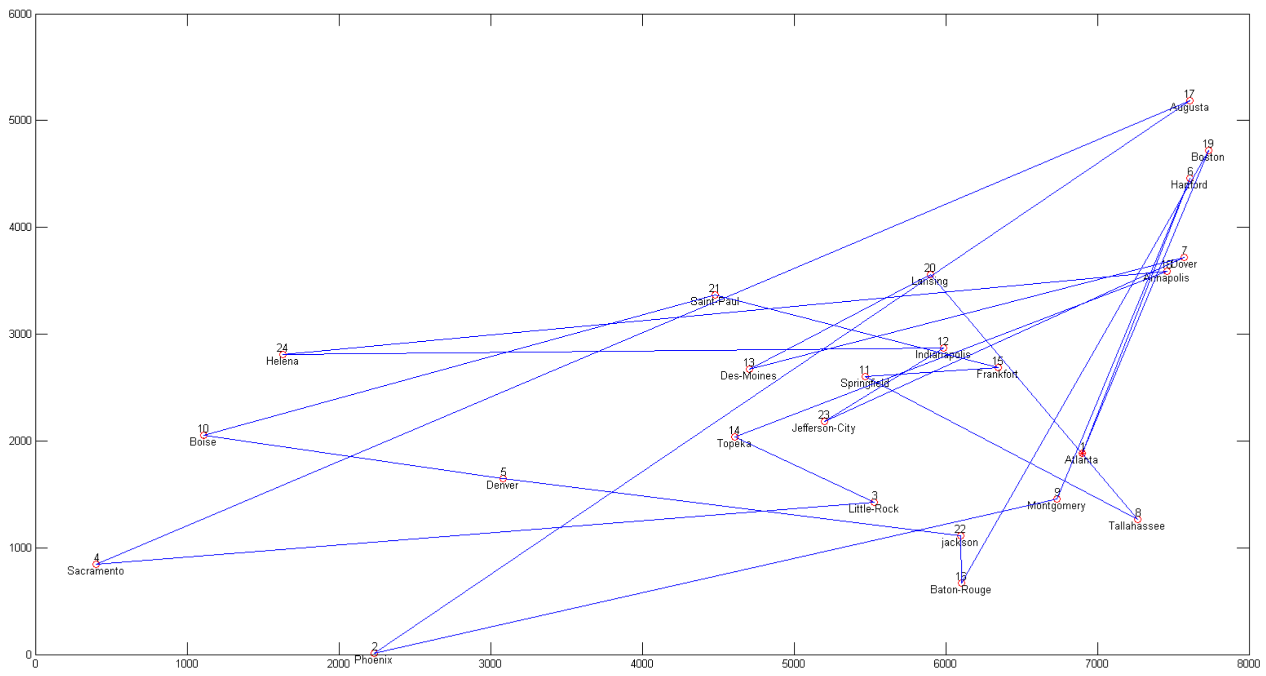

Figure 11: TSP Tour Plot - 24 City Tour with 4-City initial Sub-tour constraints

2 Source Code

The following is the source code for file $TSP-DFJ-partial.mos$ for the 48 city tour with 3-City and 4-City small sub-tour initial constraints commented out.

2.1 TSP-DFJ-partial.mos

```
model ModelName
uses "mmxprs"; !gain access to the Xpress-Optimizer solver
uses "mmsystem" ! include package to operating systems

N := 48  ! number of cities

declarations
Cities = 1 .. N                          ! set of cities
coord: array(Cities,1..3) of real ! array of coordinates of cities, to be read from US48
dist: array(Cities,Cities) of real   ! distance between each pair of cities
x : array(Cities,Cities) of mpvar        ! decision variables
flag : integer                           ! flag=0: not optimal yet; flag=1: optimal
ind : range                              ! dynamic range
numSubtour : integer                     ! number of generated subtours
numSubtourCities : integer ! number of cities on a generated subtour
SubtourCities : array(Cities) of integer ! SubtourCities(i)=1 means city i is on the sub
subtourCtr : dynamic array(ind) of linctr   ! dynamic array of subtour elimination const
TotalDist : linctr     ! objective constraint

! constraint for only one path out of each city
leavingConstraint: array(Cities)  of linctr
! constraint for only one path into of each city
enteringConstraint: array(Cities)  of linctr
! constraints for preventing one city subtour
oneCitySubTourConstraint: array(Cities)  of linctr
! constraints for preventing two city subtour
twoCitySubTourConstraint: dynamic array(range)  of linctr
! constraints for preventing three city subtour
```

```
threeCitySubTourConstraint: dynamic array(range)  of linctr
! constraints for preventing four city subtour
fourCitySubTourConstraint: dynamic array(range)  of linctr

! constraint for a TSP tour to have N edges
!tspConstr: linctr

! counter for dynamic arrays
cons: integer

! time variables
starttime: real

! keep track of next city for each city
nextCity: array(Cities) of integer

! keep the set of cities in the subtour, used to get the smallest subtour
! in a aolution
smallestSubTourSet, allSubTourCitiesSet, new_tour: set of integer

end-declarations

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!  save the tour to output file for plotting !!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! record initial time
starttime:=gettime

fopen("l_US"+N+".output",F_OUTPUT)
writeln("Starting at time :",starttime)
fclose(F_OUTPUT)

! initialization part is given
initializations from "US"+N+".dat"
     coord
end-initializations

! compute dist(i,j) the distance between each pair of cities using (x,y)
! coordinates of the cities, which are in the array coord
! you may need square root function sqrt()
```

```
!!!!!!!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!!!!!!
forall ( i in Cities ) do
forall ( j in Cities ) do
if ( i = j) then
dist(i,j) := 0.0
else
dist(i,j) := sqrt( (coord(i,2)-coord(j,2))^2 + (coord(i,3)-coord(j,3))^2 )
dist(j,i) := dist(i,j)
end-if
end-do
end-do

!!!!!!!!!!!!! objective: total distance of a tour
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
TotalDist := sum(i in Cities, j in Cities ) x(i,j)*dist(i,j)

!!!!!!!!!! write constraint: x(i,j) is binary !!!!!!!!!!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
forall(i in Cities, j in Cities ) do
x(i,j) is_binary
end-do

!!!!!!!!!!! write assignment constraints: in and out constraints for each city !!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
forall(i in Cities) do
 leavingConstraint(i) := sum ( j in Cities ) x(i,j) = 1
end-do

forall(j in Cities) do
 enteringConstraint(j) := sum ( i in Cities ) x(i,j) = 1
end-do

! 1.
!!!!!!!!!! write 1-city subtour elimination constraints here !!!!!!!!!!!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
! generate the 48c2 combinations and add the constraint for each
! combination

forall(i in Cities) do
 ! add the no closed loop constraint for each 1 city combination
 oneCitySubTourConstraint(i) := x(i,i) = 0
```

```
end-do


!2.
!!!!!!!!!!! write 2-city subtour elimination constraints here !!!!!!!!!!!!!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
cons := 1
! generate the 48c2 combinations and add the constraint for each
! combination
forall(i in Cities, j in Cities) do

 if (i>=j) then
   next
 end-if

 create(twoCitySubTourConstraint(cons))

 ! add the no closed loop constraint for each 2 city combination
 twoCitySubTourConstraint(cons) := x(i,j) + x(j,i ) <= 1

 cons += 1
end-do


! 3.
!!!!!!!!!! write 3-city subtour elimination constraints here !!!!!!!!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!!
(!
cons := 1
! generate the 48c3 combinations and add the constraint for each
! combination
forall(i in Cities, j in Cities, k in Cities ) do

 if (( i>=j) or ( i>=k) or (j>=k) ) then
  next
 end-if

 create(threeCitySubTourConstraint(cons))

 ! add the no closed loop constraint for each 3 city combination
 threeCitySubTourConstraint(cons) :=  x(i,j) + x(i,k ) + x(j,k) +
  x(j,i) + x(k,i ) + x(k,j) <= 2

 cons += 1
end-do

!4.
!!!!!!!!!! write 4-city subtour elimination constraints here !!!!!!!!!!!!!!
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!
cons := 1

! generate the 48c4 combinations and add the constraint for each
! combination
forall(i in Cities, j in Cities, k in Cities, l in Cities ) do

 if (( i>=j) or ( i>=k) or (i>=l) or (j>=k) or ( j>=l) or ( k>=l) ) then
  next
 end-if

 create(fourCitySubTourConstraint(cons))

 ! add the no closed loop constraint for each 4 city combination
 fourCitySubTourConstraint(cons) :=  x(i,j) + x(i,k ) + x(i,l) +
   x(j,i) + x(k,i ) + x(l,i) +
   x(j,k) + x(j,l) +
   x(k,j) + x(l,j) +
   x(k,l) +
   x(l,k) <= 3

  cons += 1
end-do
!)

!!!!!!!!!!!!! constraint generation algorithm !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
numSubtour := 0   ! number of added subtour elimination constraints is zero
flag := 0 ! initalize flag to be 0, so no optimal solution has been found yet

repeat

!!!!!!!!!!!!!!! Solve the restricted master problem  !!!!!!!!!!
minimize(TotalDist)
```

```
! Output the solution of the restricted master problem
writeln("The restricted master problem is solved:")
forall (i in Cities, j in Cities) do
if abs(getsol(x(i,j))-1)<0.1 then
! note here we could have simply written "if getsol(x(i,j))=1 then",
! but I found cases where Xpress doesn't output all such x(i,j)'s.
! So this is a quick and ugly fix.
! You can use this trick in the later part when you need to check if x(i,j) is 1 or not
! Also, feel free to develop your own solution
writeln("x(",i,",",j,")=",getsol(x(i,j)))
 ! save the next city information for each city in array
next_city(i) := j
end-if
end-do

!!!!!!!!!!!!!!!!!!!!!  find a subtour !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! We want to find a subtour starting at city 1 (Atlanta) and ending at City 1 (such a subtour
! First, initialize a few things:
numSubtourCities := 0    ! the number of cities on the subtour
forall (i in Cities) do  ! SubtourCities(i)=1 if city i is on the subtour, initialize all entr
SubtourCities(i):=0  ! need to change entries when city i is found on the tour
end-do
SubtourCities(1) := 1  ! City 1 (Atlanta) is always on the subtour

! Start the procedure to look for a subtour starting and ending at City 1.
! The basic algorithm is discussed in the hand-out
! Note you need to update SubtourCities for cities that are on the subtour
! You also need to keep track of the number of cities numSubtourCities on the subtour
!!!!!!!!! fill in your code here !!!!!!!!!!!!!!

! loop from atlanta till we reach atlanta back again
currentCity := 1
repeat
! set of cities in this subtour
smallestSubTourSet += {currentCity}

! find the next city j for TSP from this city i
currentCity := next_city(currentCity)
numSubtourCities += 1
SubtourCities(currentCity) := 1
```

```
until ( currentCity = 1 )

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! output the subtour you found
writeln("Found a subtour of distance ", getobjval, " and ", numSubtourCities, " cities")
writeln("Cities on the subtour are:")
forall (i in Cities | SubtourCities(i) = 1) do
! Note: forall ( ... | express ) is very useful, you may need to use it in the followin
! part to add subtour elimination constraints
write(i, " ")
end-do
writeln("")

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!  save the tour to output file for plotting !!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
fopen("l_US"+N+".output",F_OUTPUT+F_APPEND)
writeln("Constraints Added : ",numSubtour)
writeln("Time in Secs : ", gettime-starttime)
writeln("Objective Distaince : ", getobjval)
writeln("Subtour from Atlanta : ", numSubtourCities)
writeln("Full Tour:")
forall (i in Cities) do
writeln(i,"\t",next_city(i))
end-do
writeln("-------------------------")
fclose(F_OUTPUT)

!!!!!!!!!!!!!!!!!!!! add the subtour elimination constraint !!!!!!!!!!!!!!!!!!!!!!!!!!!
! If the subtour found above is indeed a subtour (i.e. has fewer than 48 cities),
! then add the corresponding subtour elimination
!  constraint to the problem
! otherwise, if the subtour has 48 cities, then it’s a TSP tour and optimal,
! terminate the constraint generation by setting the flag to 1
!!!!!! fill in you code !!!!!!!!!!

if ( numSubtourCities = N ) then
   flag := 1
else
```

```
! find the smallest subtour in the solution and a constraint to break it

! only if the subtout with city 1 is > 1, else it has to be the smallest size
if getsize(smallestSubTourSet) > 1 then

! set to keep track of cities in subtours xconsidered so far,
! initialized to the set of cities in subtour with city 1
allSubTourCitiesSet := smallestSubTourSet

! go over the cities not in subtours considered so far and find a subtour for each
forall (i in Cities) do

 ! if the city is not in the subtour considered so far, then find the subtour
 ! having this city
 if ( i not in allSubTourCitiesSet) then

! the city is not in any subtour so far, find the subtour with this city
new_tour := {}
currentCity := i
repeat
new_tour += {currentCity}
until ( currentCity = i )

! add the cities in this subtour to the cities in subtour so far
allSubTourCitiesSet += new_tour

! if this tour is smaller than the earlier one  then save this as the smallest
! subtour in this solution
if ( getsize(new_tour) < getsize(smallestSubTourSet) ) then
smallestSubTourSet := new_tour
end-if

! if this smallest subtour is 1 then any subtour cannot be smaller than this
! subtour in this solution
if ( getsize(new_tour) = 1 ) then
smallestSubTourSet := new_tour
break
end-if

 end-if
```

```
end-do

end-if

! add a constraint to break the smallest Sub Tour found
numSubtour += 1
create(subtourCtr(numSubtour))
subtourCtr(numSubtour):= sum (i in smallestSubTourSet ) x(i, next_city(i)) <= getsize(sm

end-if

    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


until flag = 1

!!!!!!!!!!!!!!!!!!!! end of the constraint generation algorithm !!!!!!!!!!!!!!!!!!!!!!!

writeln("\nOptimal TSP distance = ", getobjval)
forall (i in Cities, j in Cities) do
if abs(getsol(x(i,j))-1)<0.1 then
writeln("x(",i,",",j,")=",getsol(x(i,j)))
end-if
end-do

! write the solution to an output file
! then run matlab code US48TourPlot.m to plot the tour
fopen("US"+N+".output",F_OUTPUT)
forall (i in Cities, j in Cities) do
if abs(getsol(x(i,j))-1)<0.1 then
writeln(i,"\t",j)
end-if
end-do
fclose(F_OUTPUT)

writeln("End running model")

end-model
```

2.2 US24TourPlot.m

The following is the source code for $US24TourPlot.m$, for plotting the TSP for the 24 City tour.

```
clear all;

outputfile = 'US24.output';
outputfile = 'US24.output';
f = fopen(outputfile,'r');
x = fscanf(f, '%d\t %d', [2, inf]);
fclose(f);
x = x';

 read the list of cities
cities = cell(1e6, 2) ;
citiesfile = 'cities.input';
f = fopen(citiesfile,'r');
rCnt = 0 ;
while ~feof(f)
    rCnt = rCnt + 1 ;
    cities{rCnt,1} = fscanf(f, '%d', 1) ;
    cities{rCnt,2} = fscanf(f, '%s', 1) ;
end
fclose(f);
cities = cities(1:rCnt-1,:) ;

coordfile = 'US48.input';
f = fopen(coordfile,'r');
coord = fscanf(f, '%d %f %f', [3, inf]);
fclose(f);
coord = coord';

tour = zeros(24,1);
tour(1) = 1;
fromCity = 1;
for k = 1 : 24
    tour(k+1) = x(fromCity,2);
    fromCity = x(fromCity,2);
end
figure(1);
```

```
plot(coord(tour(:,1),2),coord(tour(:,1),3),'rO');
text(coord(tour(:,1),2),coord(tour(:,1),3),num2str(coord(tour(:,1),1)),'HorizontalAlignm
text(coord(tour(:,1),2),coord(tour(:,1),3), cities(tour(:,1),2),'HorizontalAlignment','c


hold on;
plot([coord(tour(:,1),2);coord(1,2)],[coord(tour(:,1),3);coord(1,3)]);
plot(coord(1,2),coord(1,3),'r*'); % the star marks Atlanta
```

3 Distribution of Team Effort

Equal effort by all team members. Each team member implemented the solutions independently and verified the results among the team.

4 References

[1] Bertsimas Dimitris, Tsitsiklis N. John, *Introduction to Linear Optimization*, Athena Scientific Edition 6, 1997.