MAT 168 Modeling 3

Hardy Jones
999397426
Professor Köppe
Spring 2015

3. (a) After modifying the given ZIMPL file and running it on all 14 tours, we can compare to the optimal:

Tour	Optimal	Nearest Neighbor	Subtours
a280	2579	3157	2550
ali535	202310	265685	193429
berlin52	7542	8980	7164
burma14	3323	4501	3001
gr137	69853	98781	67009
gr202	40160	54092	38576
gr229	134602	162109	128353
gr431	171414	206628	163905
gr666	294358	364429	286428
gr96	55209	74939	53069
pr226	80369	94683	57177
u574	36905	50459	34422
ulysses16	6859	8081	6113
ulysses22	7013	8248	6160

What we see is that every solution is more optimal than the optimal solution.

(b) See Figure 1

(c) After changing the variables to real values between 0 and 1, we now have these results:

Tour	Optimal	Nearest Neighbor	Subtours	Subtours Fractional
a280	2579	3157	2550	2534
ali535	202310	265685	193429	191454.5
berlin52	7542	8980	7164	7163
burma14	3323	4501	3001	3001
gr137	69853	98781	67009	66643.5
gr202	40160	54092	38576	38383.5
gr229	134602	162109	128353	127411
gr431	171414	206628	163905	163027
gr666	294358	364429	286428	284803.5
gr96	55209	74939	53069	52728.5
pr226	80369	94683	57177	55247.5
u574	36905	50459	34422	34256
ulysses16	6859	8081	6113	6113
ulysses22	7013	8248	6160	6106.5

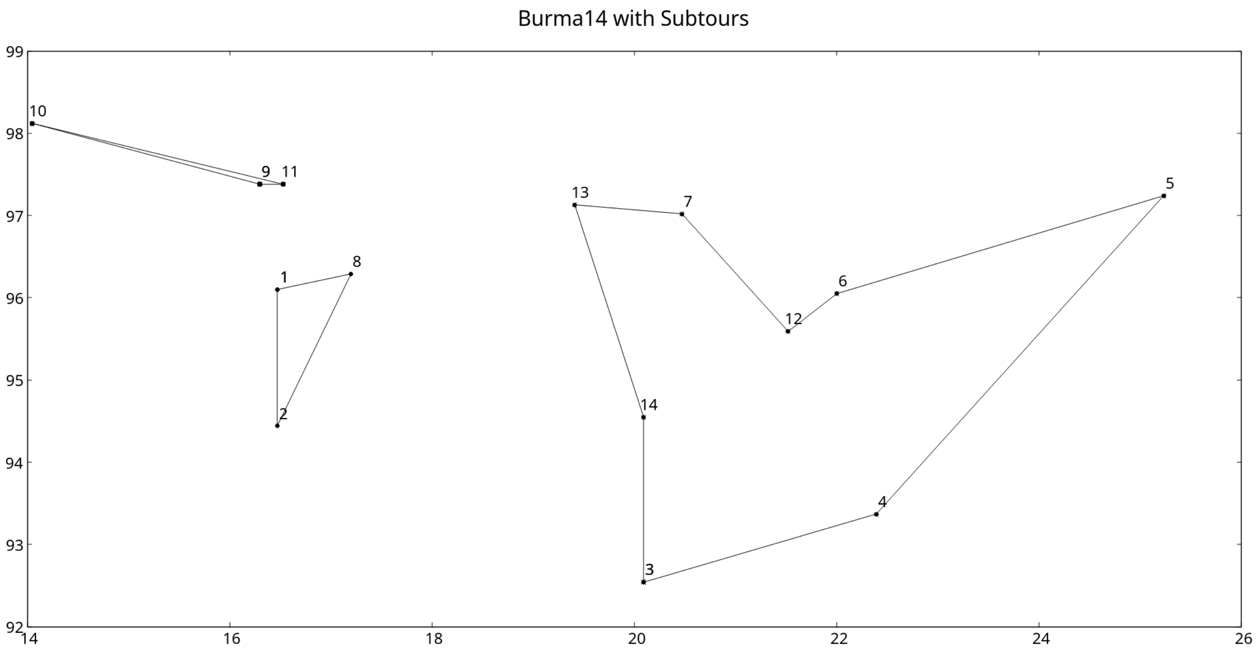

Figure 1: Visualization of `burma14Distances.txt`

The first thing to notice is that each example runs much quicker. Since we have relaxed the problem, this should not be a surprise.

We also see that most of the tours now have slightly more optimal solutions than before.

4. (a) After modifying the model to eliminate subtours, only two tours can be computed. The other tours consume too much memory to complete and are killed by the OS.

   The two tours are `burma14` and `ulysses16`–with `ulysses16` being the larger tour. And both of these tours have the optimal solution of 3323 and 6859 respectively. See Figure 2

   (b) Again, we can only run the two tours `burma14` and `ulysses16`.

   Interestingly the tours generated are the same, so we eschew visualizing the same tours.

5. Yet again we are restricted to `burma14` and `ulysses16`. In each of the cases, what we find is that the k doesn't make a difference on the outcome.

   We can add this information to the table.

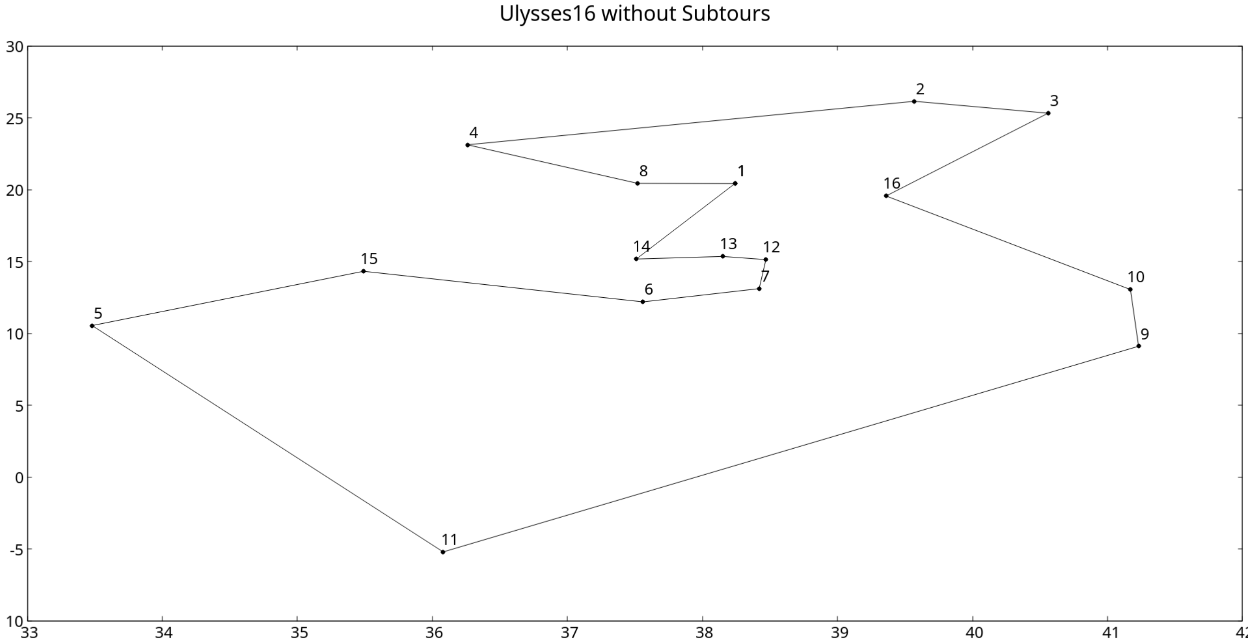

Figure 2: Visualization of `ulysses16` without subtours

Tour	Opt.	N. N.	Subtours	Subtours Fractional	$k \in \{3, 4, 5\}$
a280	2579	3157	2550	2534	N/A
ali535	202310	265685	193429	191454.5	N/A
berlin52	7542	8980	7164	7163	N/A
burma14	3323	4501	3001	3001	3098
gr137	69853	98781	67009	66643.5	N/A
gr202	40160	54092	38576	38383.5	N/A
gr229	134602	162109	128353	127411	N/A
gr431	171414	206628	163905	163027	N/A
gr666	294358	364429	286428	284803.5	N/A
gr96	55209	74939	53069	52728.5	N/A
pr226	80369	94683	57177	55247.5	N/A
u574	36905	50459	34422	34256	N/A
ulysses16	6859	8081	6113	6113	6228
ulysses22	7013	8248	6160	6106.5	N/A

For the sake of brevity we only visualize one tour.

See Figure 3

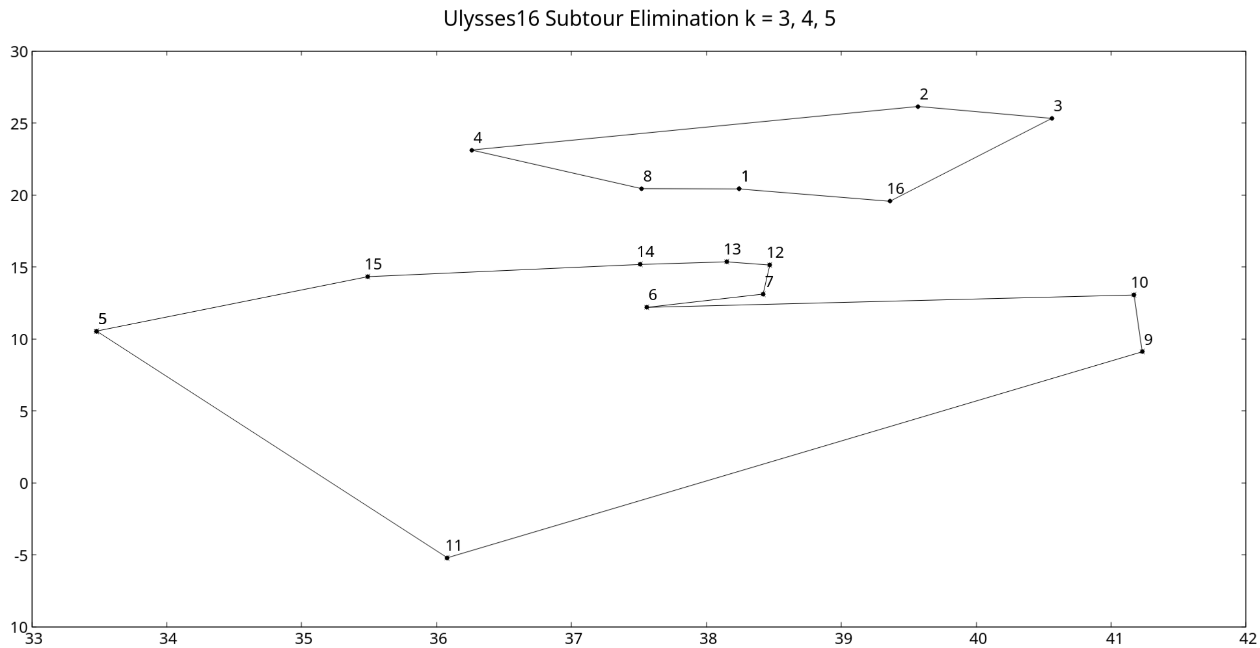

Figure 3: Visualization of `ulysses16` without subtours for k = 3, 4, 5

Appendix A ZIMPL

For each of the modeling files, we use a simple shell script to run them.

This allows a decent interface that allows injecting a filename without manually manipulating the files. Each is a slight modification of the following format.

```
#!/usr/bin/env sh

files() {
  echo "1  a280"
  echo "2  ali535"
  echo "3  berlin52"
  echo "4  burma14"
  echo "5  gr137"
  echo "6  gr202"
  echo "7  gr229"
  echo "8  gr431"
  echo "9  gr666"
  echo "10 gr96"
  echo "11 pr226"
  echo "12 u574"
  echo "13 ulysses16"
  echo "14 ulysses22"
}

name() {
  case $1 in
    1  ) echo "a280Distance.txt" ;;
    2  ) echo "ali535Distances.txt" ;;
    3  ) echo "berlin52Distance.txt" ;;
    4  ) echo "burma14Distances.txt" ;;
    5  ) echo "gr137Distances.txt" ;;
    6  ) echo "gr202Distances.txt" ;;
    7  ) echo "gr229Distances.txt" ;;
    8  ) echo "gr431Distances.txt" ;;
    9  ) echo "gr666Distances.txt" ;;
```

```
    10 ) echo "gr96Distances.txt" ;;
    11 ) echo "pr226Distance.txt" ;;
    12 ) echo "u574Distance.txt" ;;
    13 ) echo "ulysses16Distances.txt" ;;
    14 ) echo "ulysses22Distances.txt" ;;
  esac
}

tour() {
  case $1 in
    1  ) echo "a280" ;;
    2  ) echo "ali535" ;;
    3  ) echo "berlin52" ;;
    4  ) echo "burma14" ;;
    5  ) echo "gr137" ;;
    6  ) echo "gr202" ;;
    7  ) echo "gr229" ;;
    8  ) echo "gr431" ;;
    9  ) echo "gr666" ;;
    10 ) echo "gr96" ;;
    11 ) echo "pr226" ;;
    12 ) echo "u574" ;;
    13 ) echo "ulysses16" ;;
    14 ) echo "ulysses22" ;;
  esac
}

while true; do
  files
  read -p "Choose a file number or <q> to quit: " num
  if (( 1 <= num && num <= 14 )); then
    sed "s/PUT_THE_FILENAME_HERE/\"$(name $num)\"/" subtours_0_1.tmpl.zpl > subtours_0_1.zpl
    scip -f subtours_0_1.zpl > "$(tour $num)_0_1.log"
  elif [[ "$num" == q* ]]; then
    exit
  else
    echo "That is not valid"
  fi
  echo
done
```

For problem 3a we use the following model:

```
# Beginning of a TSP model

# Filename
param name := PUT_THE_FILENAME_HERE;

# Number of cities
param n := read name as "1n" use 1;

set V := { 1..n };

set E := { <i, j> in V cross V with i < j };

# Edge variables
var x[E] binary;

# Distances
param d[E] := read name as "<1n, 2n> 3n" skip 1;

minimize tour_length:
  sum <i,j> in E : d[i,j] * x[i,j];

subto degree:
  forall <v> in V do
    sum <v,j> in E : x[v,j]  +  sum <i,v> in E : x[i,v] == 2;
```

For problem 3c we use the following model:

```
# Beginning of a TSP model

# Filename
param name := PUT_THE_FILENAME_HERE;

# Number of cities
param n := read name as "1n" use 1;

set V := { 1..n };

set E := { <i, j> in V cross V with i < j };

# Edge variables
var x[E] real <= 1;

# Distances
param d[E] := read name as "<1n, 2n> 3n" skip 1;

minimize tour_length:
  sum <i,j> in E : d[i,j] * x[i,j];

subto degree:
  forall <v> in V do
    sum <v,j> in E : x[v,j]  +  sum <i,v> in E : x[i,v] == 2;
```

For problem 4a we use the following model:

```
# Beginning of a TSP model

# Filename
param name := PUT_THE_FILENAME_HERE;

# Number of cities
param n := read name as "1n" use 1;

set V := { 1..n };

set E := { <i, j> in V cross V with i < j };

# Edge variables
var x[E] binary;

# Distances
param d[E] := read name as "<1n, 2n> 3n" skip 1;

minimize tour_length:
  sum <i,j> in E : d[i,j] * x[i,j];

subto degree:
  forall <v> in V do
    sum <v,j> in E : x[v,j]  +  sum <i,v> in E : x[i,v] == 2;

# Subtour elimination:

set S[] := powerset(V);

set S_Indices := indexset(S);

subto no_subtour:

  forall <s_index> in S_Indices with
    card(S[s_index]) >= 3 and card(S[s_index]) <= n - 3 do

      sum <i,j> in E with <i> in S[s_index] and <j> in S[s_index] :
        x[i,j] <= card(S[s_index]) - 1;
```

For problem 4b we use the following model:

```
# Beginning of a TSP model

# Filename
param name := PUT_THE_FILENAME_HERE;

# Number of cities
param n := read name as "1n" use 1;

set V := { 1..n };

set E := { <i, j> in V cross V with i < j };

# Edge variables
var x[E] real <= 1;

# Distances
param d[E] := read name as "<1n, 2n> 3n" skip 1;

minimize tour_length:
  sum <i,j> in E : d[i,j] * x[i,j];

subto degree:
   forall <v> in V do
     sum <v,j> in E : x[v,j]  +  sum <i,v> in E : x[i,v] == 2;

# Subtour elimination:

set S[] := powerset(V);

set S_Indices := indexset(S);

subto no_subtour:

   forall <s_index> in S_Indices with
     card(S[s_index]) >= 3 and card(S[s_index]) <= n - 3 do

       sum <i,j> in E with <i> in S[s_index] and <j> in S[s_index] :
         x[i,j] <= card(S[s_index]) - 1;
```

For problem 5 we use the following model:

```
# Beginning of a TSP model

# Filename
param name := PUT_THE_FILENAME_HERE;

# K
param k := 3;

# Number of cities
param n := read name as "1n" use 1;

set V := { 1..n };

set E := { <i, j> in V cross V with i < j };

# Edge variables
var x[E] real <= 1;

# Distances
param d[E] := read name as "<1n, 2n> 3n" skip 1;

minimize tour_length:
  sum <i,j> in E : d[i,j] * x[i,j];

subto degree:
  forall <v> in V do
    sum <v,j> in E : x[v,j]  +  sum <i,v> in E : x[i,v] == 2;

# Subtour elimination:

set S[] := powerset(V);

set S_Indices := indexset(S);

subto no_subtour:

  forall <s_index> in S_Indices with
    card(S[s_index]) >= 3 and card(S[s_index]) <= k do

      sum <i,j> in E with <i> in S[s_index] and <j> in S[s_index] :
        x[i,j] <= card(S[s_index]) - 1;
```

For sake of brevity, we just show one model for $k = 3$. The other models are similar with the value of k changed.