Team 1 Project Model

Duane Murray

11/17/2021

Miller-Tucker-Zemlin (MTZ) formulation for Traveling Salesperson Problem (TSP)

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij}, && (1)\\
\text{subject to} & && (2)\\
& \sum_{i=1, i \neq j}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n, && (3)\\
& \sum_{j=1, j \neq i}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, && (4)\\
& u_i - u_j + n x_{ij} \leq n - 1, \quad 2 \leq i \neq j \leq n, && (5)\\
& x_{ij} \in \{0, 1\} \quad i, j = 1, 2, \ldots, n, \quad i \neq j, && (6)\\
& u_i \in \mathbb{R}^{+} \quad i = 1, 2, \ldots, n. && (7)
\end{aligned}
$$

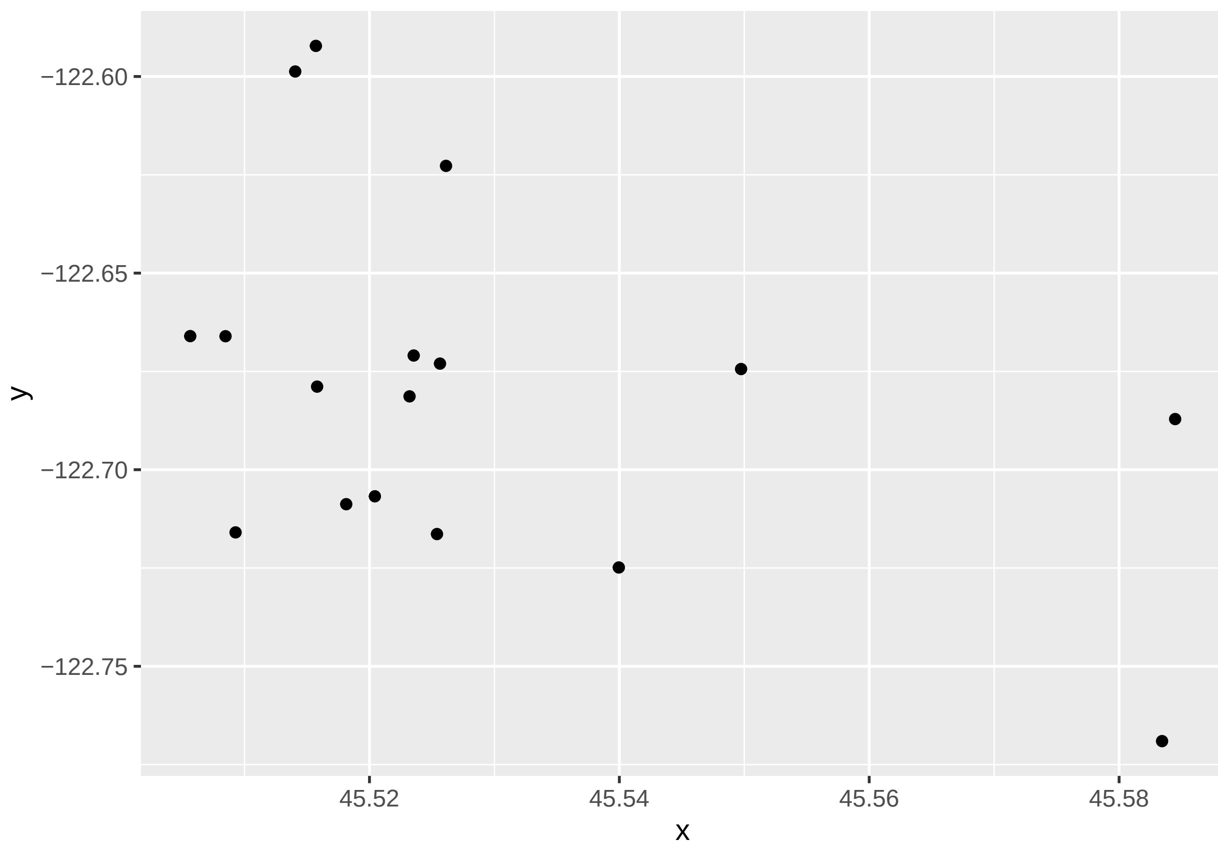
Base Traveling Salesman Problem ompr Model Code to Work From

```r
setwd("G:/My Drive/FALL-2021/ETM640/Project/Code/") # SET WORKING DIR

refined_locations <- read.csv("TEST_portland_location_data.csv") # LOAD DATA FROM FILE

n <- nrow(refined_locations) # NUMBER OF LOCATIONS TO VISIT (replace with number of data matrix rows)
#n <- 10

# from 0 to ...
#max_x <- 500
#max_y <- 500
#set.seed(2451)

locations <- data.frame(id = 1:n, x = refined_locations[,8], y = refined_locations[,9])
#cities <- data.frame(id = 1:n, x = runif(n, max = max_x), y = runif(n, max = max_y))

pander(locations)
```

id	x	y
1	45.52	-122.7
2	45.52	-122.7
3	45.51	-122.7
4	45.53	-122.7

id	x	y
5	45.52	-122.7
6	45.51	-122.7
7	45.53	-122.7
8	45.54	-122.7
9	45.52	-122.7
10	45.52	-122.6
11	45.51	-122.6
12	45.58	-122.8
13	45.53	-122.6
14	45.51	-122.7
15	45.58	-122.7
16	45.52	-122.7
17	45.55	-122.7

```
ggplot(locations, aes(x, y)) +
  geom_point()
```




```
distance <- as.matrix(stats::dist(select(locations, x, y), diag = TRUE, upper = TRUE))
dist_fun <- function(i, j) {
  vapply(seq_along(i), function(k) distance[i[k], j[k]], numeric(1L))
}

# MIPModel() is standard method
# MILPmodel() is beta, and purported as 1000 times faster than MIP
```

```r
model <- MIPModel() %>%
  # we create a variable that is 1 iff we travel from location i to j
  add_variable(x[i, j], i = 1:n, j = 1:n,
               type = "integer", lb = 0, ub = 1) %>%
  # a helper variable for the MTZ formulation of the TSP
  add_variable(u[i], i = 1:n, lb = 1, ub = n) %>%
  # minimize travel distance
  set_objective(sum_expr(dist_fun(i, j) * x[i, j], i = 1:n, j = 1:n), "min") %>%
  # you cannot go to the same location
  set_bounds(x[i, i], ub = 0, i = 1:n) %>%
  # leave each location
  add_constraint(sum_expr(x[i, j], j = 1:n) == 1, i = 1:n) %>%
  # visit each location
  add_constraint(sum_expr(x[i, j], i = 1:n) == 1, j = 1:n) %>%
  # ensure no sub-tours are used (arc constraints)
  add_constraint(u[i] >= 2, i = 2:n) %>%
  add_constraint(u[i] - u[j] + 1 <= (n - 1) * (1 - x[i, j]), i = 2:n, j = 2:n)

result <- solve_model(model, with_ROI(solver = "glpk", verbose = TRUE))
```

```
## <SOLVER MSG>  ----
## GLPK Simplex Optimizer, v4.47
## 306 rows, 306 columns, 1330 non-zeros
##       0: obj =  0.000000000e+000  infeas = 5.000e+001 (34)
## *    54: obj =  9.301889643e-001  infeas = 0.000e+000 (1)
## *   114: obj =  3.347211881e-001  infeas = 0.000e+000 (1)
## OPTIMAL SOLUTION FOUND
## GLPK Integer Optimizer, v4.47
## 306 rows, 306 columns, 1330 non-zeros
## 289 integer variables, 272 of which are binary
## Integer optimization begins...
## +   114: mip =     not found yet >=              -inf        (1; 0)
## +   479: >>>>>  5.612905937e-001 >=  3.411718128e-001  39.2% (53; 0)
## +   760: >>>>>  5.501492966e-001 >=  3.439731575e-001  37.5% (81; 2)
## +  2374: >>>>>  5.490180214e-001 >=  3.486128488e-001  36.5% (207; 14)
## +  5552: >>>>>  4.886910076e-001 >=  3.569555982e-001  27.0% (454; 30)
## + 13773: >>>>>  4.583297546e-001 >=  3.875491625e-001  15.4% (979; 311)
## + 28543: >>>>>  4.577475636e-001 >=  3.999576764e-001  12.6% (1575; 1152)
## +126374: mip =  4.577475636e-001 >=  4.186957891e-001   8.5% (6497; 4516)
## +224767: mip =  4.577475636e-001 >=  4.293303834e-001   6.2% (9033; 9337)
## Warning: numerical instability (dual simplex, phase II)
## +308947: mip =  4.577475636e-001 >=  4.391175831e-001   4.1% (9965; 14818)
## +383636: mip =  4.577475636e-001 >=  4.453488926e-001   2.7% (9236; 21753)
## +452664: mip =  4.577475636e-001 >=  4.508290076e-001   1.5% (5978; 32952)
## +500926: mip =  4.577475636e-001 >=     tree is empty   0.0% (0; 58661)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

```r
solution <- get_solution(result, x[i, j]) %>%
  filter(value > 0)
kable(head(solution, 3))
```

variable	i	j	value
x	2	1	1
x	3	2	1
x	4	3	1

```r
paths <- select(solution, i, j) %>%
  rename(from = i, to = j) %>%
  mutate(trip_id = row_number()) %>%
  tidyr::gather(property, idx_val, from:to) %>%
  mutate(idx_val = as.integer(idx_val)) %>%
  inner_join(locations, by = c("idx_val" = "id"))
kable(head(arrange(paths, trip_id), 4))
```

trip_id	property	idx_val	x	y
1	from	2	45.51814	-122.7088
1	to	1	45.52043	-122.7068
2	from	3	45.50928	-122.7159
2	to	2	45.51814	-122.7088

```r
ggplot(locations, aes(x, y)) +
  geom_point() +
  geom_line(data = paths, aes(group = trip_id)) +
  ggtitle(paste0("Optimal route with cost: ", round(objective_value(result), 2)))
```