Team 1 Project Model

Duane Murray

11/17/2021

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

$$\min \quad \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} c_{ij} x_{ij}, \tag{1}$$

$$\text{subject to} \tag{2}$$

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1, \quad j = 1, 2, \ldots, n, \tag{3}$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1, \quad i = 1, 2, \ldots, n, \tag{4}$$

$$u_i - u_j + n x_{ij} \leq n - 1, \quad 2 \leq i \neq j \leq n, \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, 2, \ldots, n, \quad i \neq j, \tag{6}$$

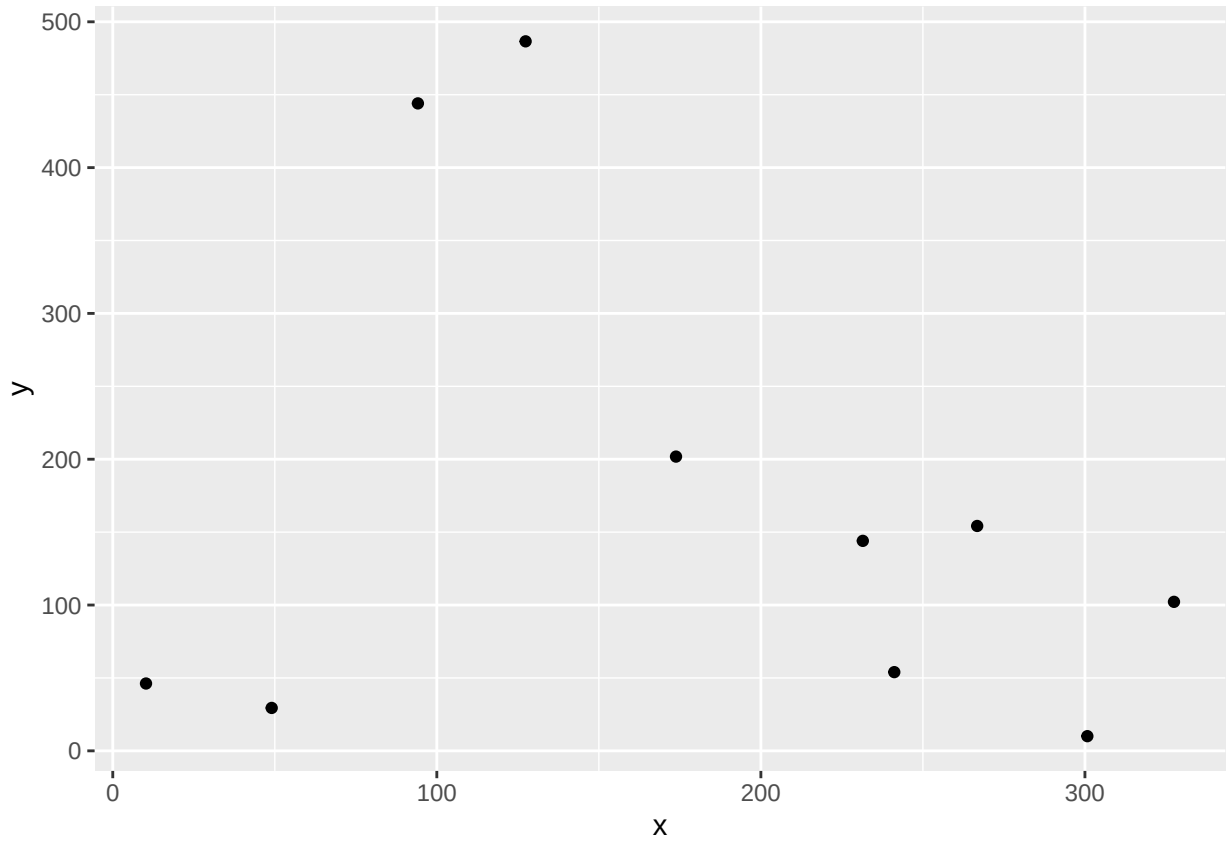
$$u_i \in \mathbb{R}^{+} \quad i = 1, 2, \ldots, n. \tag{7}$$

Base Traveling Salesman Problem ompr Model Code to Work From

```
n <- 10 # NUMBER OF LOCATIONS TO VISIT (replace with number of data matrix rows)
# from 0 to ...
max_x <- 500
max_y <- 500

set.seed(2451)
cities <- data.frame(id = 1:n, x = runif(n, max = max_x), y = runif(n, max = max_y))

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




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


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
## 110 rows, 110 columns, 434 non-zeros
##       0: obj =  0.000000000e+000  infeas = 2.900e+001 (20)
## *    33: obj =  2.577959235e+003  infeas = 0.000e+000 (1)
## *    72: obj =  8.032662331e+002  infeas = 0.000e+000 (1)
## OPTIMAL SOLUTION FOUND
## GLPK Integer Optimizer, v4.47
## 110 rows, 110 columns, 434 non-zeros
## 100 integer variables, 90 of which are binary
## Integer optimization begins...
## +    72: mip =     not found yet >=              -inf        (1; 0)
## +   172: >>>>>  1.521545667e+003 >=  8.145818326e+002  46.5% (12; 0)
## +   523: >>>>>  1.353374092e+003 >=  8.909793197e+002  34.2% (38; 9)
## +  2511: mip =  1.353374092e+003 >=     tree is empty   0.0% (0; 259)
## INTEGER OPTIMAL SOLUTION FOUND
## <!SOLVER MSG> ----
```

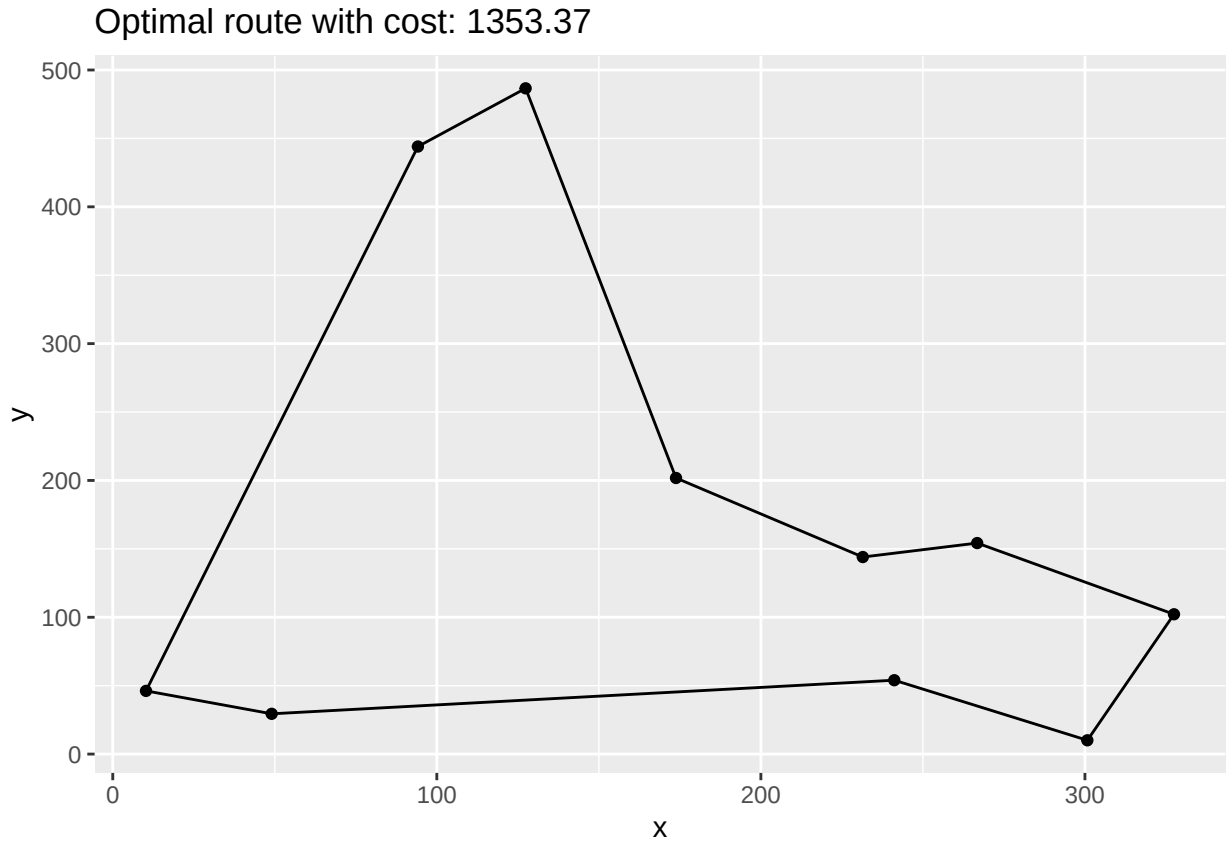
```
solution <- get_solution(result, x[i, j]) %>%
  filter(value > 0)
kable(head(solution, 3))
```

variable	i	j	value
x	7	1	1
x	10	2	1
x	1	3	1

```
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(cities, by = c("idx_val" = "id"))
kable(head(arrange(paths, trip_id), 4))
```

trip_id	property	idx_val	x	y
1	from	7	300.7493	10.07039
1	to	1	241.1684	53.98388
2	from	10	266.7549	154.22619
2	to	2	327.4863	102.19240

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


Optimal route with cost: 1353.37
500
400
300
200
100
0
y
0
100
200
300
x