JuMP
Linear Programming

Przemysław Szufel
https://szufel.pl/

Use case scenario

The Subway restaurant chain in Las Vegas has a total of 118 restaurants in different parts of the city.

18 restaurants have adjacent huge product warehouses that keep ingredients cool and fresh, moreover fresh vegetables are delivered only to those warehouses (rather than to every restaurant) daily at 3am.

Subway has signed a contract with a transportation agency and is billed by the multiple of the weight of transported goods and the distance.

Knowing the amount of available stock at each warehouse and the expected demand at each restaurant (measured in kg), the company needs to decide how the goods should be distributed among warehouses.

Transportation problem statement

- Variables
  - x_{ij} – number of units transported for i-th supplier to j-th requester
  - c_{ij} – unit transportation cost between i-th supplier to j-th requester
- Cost function C

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} \boldsymbol{c}_{ij} \boldsymbol{x}_{ij}$$

- Constraints:

  suppliers have maximum capacity S_i

$$\sum_{j=1}^{n} x_{ij} \leq S_i$$

  demand D_j must be met

$$\sum_{i=1}^{m} x_{ij} \geq D_j$$

Data preparation and preprocessing

- Subway restaurant location data: http://www.poi-factory.com/node/30506
- Selected rows for LV: **`Subway_LV.csv`**
- Problem: calculate distance matrix between restaurants
  - Shortest path optimization (Dijkstra)
  - Distributed computing in Julia

Routing algorithms

source: Delling et al. (2009). Engineering route planning algorithms. In Algorithmics of large and complex networks (pp. 117-139), Springer

method	size $n/10^6$	space [B/n]	preproc. [min]	speedup
DIJKSTRA	18	21	0	1
separator multi-level	0.1	?	$>$ 5 400	52
edge flags (basic)	1	35	299	523
landmark A^*	18	89	13	28
edge flags	18	30	1 028	3 951
HHs (basic)	18	49	161	2 645
reach + shortc. + A^*	18	100	1 625	1 559
	18	56	141	3 932
HHs + dist. tab.	18	68	13	10 364
HHs + dist. tab. + A^*	18	92	14	12 902
high-perf. multi-level	18	181	1 440	401 109
transit nodes (eco)	18	140	25	574 727
transit nodes (gen)	18	267	75	1 470 231
highway nodes	18	28	15	7 437
approx. planar $\epsilon = 0.01$	1	2 000	150	18 057
SHARC	18	34	107	21 800
bidirectional SHARC	18	41	212	97 261
contr. hier. (aggr)	18	17	32	41 051
contr. hier. (eco)	18	21	10	28 350
CH + edge flags (aggr)	18	32	99	371 882
CH + edge flags (eco)	18	20	32	143 682
transit nodes + edge flags	18	341	229	3 327 372
contr. hier. (mobile)	18	8	31	9 878

Calculating distance paths on map data

Steps

1. Acquire street system map for Las Vegas (https://www.openstreetmap.org/)
2. Represent the map data as a graph (use OpenStreetMapX.jl)
3. Calculate the shortest path for every pair of Subway restaurants in Las Vegas (for the simplicity traffic location is ignored)
4. Since a total of 13,806 distances need to be calculated, run the computations at scale

Using Dijkstra in Julia
(example – shortest path between node 2 and 6)

```
julia> using LightGraphs

julia> g = CycleGraph(10);

julia> dj = dijkstra_shortest_paths(g, 2);

julia> enumerate_paths(dj)[6]
```

Distributed computing of shortest paths between nodes (code part)

```
@everywhere function calc_distances(i,N, sbws_la)
    open(string(lpad(i,4,"0"),"_distance.csv"),"w") do f
        for j in 1:N
            dista = 0.0
            route = [sbws_la.node[i]]
            if sbws_la.node[i] != sbws_la.node[j]
                route, dista, time = shortest_route(map_data.network, sbws_la.node[i], sbws_la.node[j])
            end
            println(f,"$i,$j,$dista,$(join(route,"#"))")
        end
    end
end
all_no = @distributed (+) for i in 1:N
    calc_distances(i, N, sbws_la)
    1
end
```

Implementation in JuMP

```
m = Model(solver=GLPKSolverLP())
@variable(m, x[i=1:S, j=1:D])
@objective(m, Min, sum( x[i, j]*distance_mx[i, j] for i=1:S, j=1:D))
@constraint(m, x .>= 0)
for j=1:D
   @constraint(m, sum( x[i, j] for i=1:S) >= demand[j]   )
end
for i=1:S
   @constraint(m, sum( x[i, j] for j=1:D) <= supply[i]   )
end
status = solve(m)
getvalue(x)
```