Clubbing Multiple Waypoints together to generate smart route

Kyle L. Schutt

Wei Wang

Parang Saraf

Problem Statement

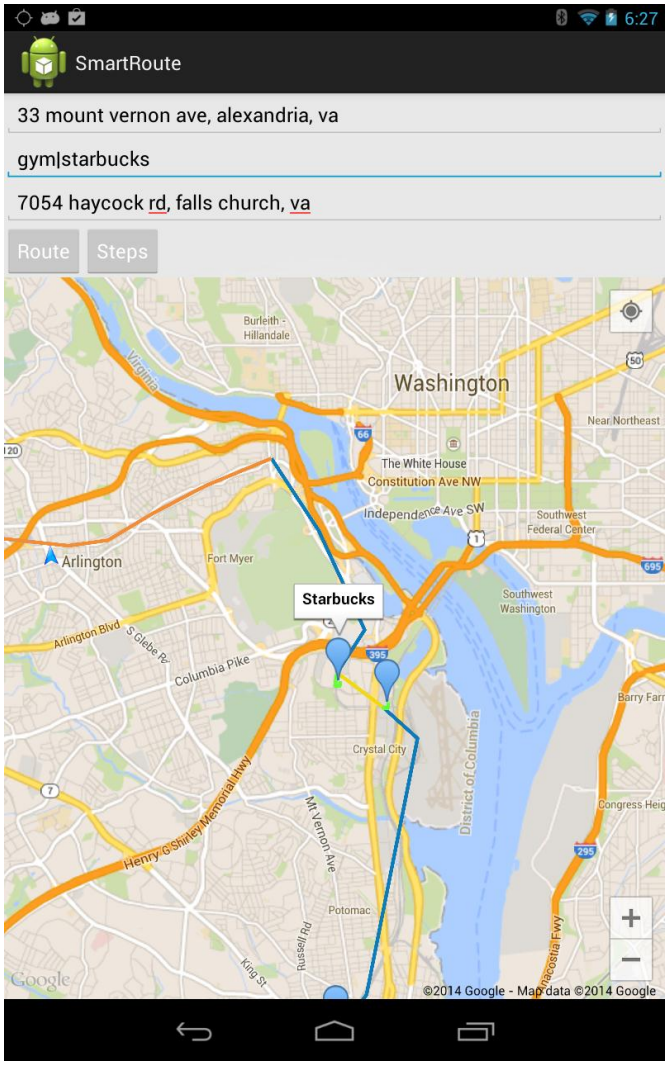
- Find a smart route using public transit, given:
  - A source
  - A destination
  - Multiple Waypoints

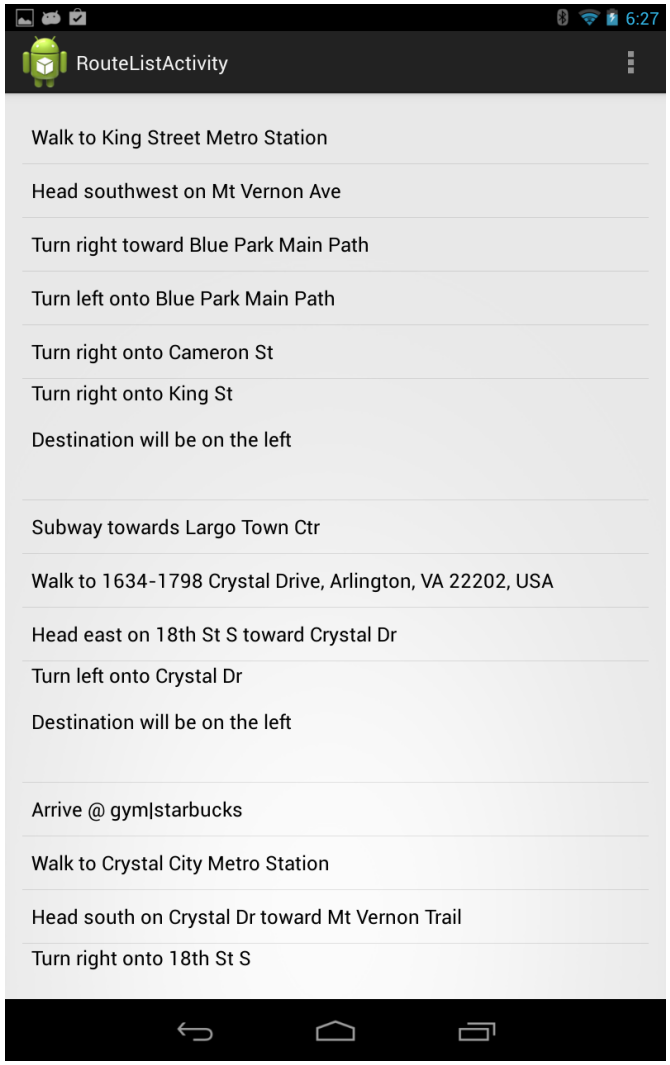
    Where waypoints can be either clubbed together or remain separate based on best possible route

Previous Work

- Google Directions
  - Doesn’t let users define waypoints for public transit options
  - No selection of “least costly” route
- Navigation Devices
  - Allows user to specify waypoints but doesn’t optimize results

Our Solution





Our Solution

- Open API from WMATA
- Google Places, Direction and Map APIs
- Custom algorithm for determining POIs
- Another way to navigate

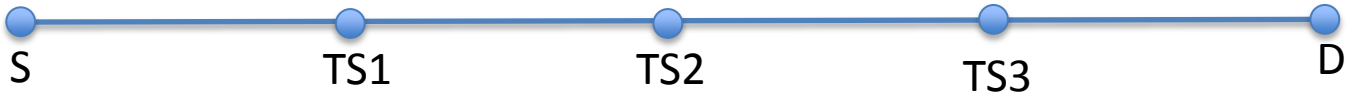
User Inputs

- User can specify
  - Source
  - Destination
  - One or two generic waypoints
- Ideally the system should return user a smart route with:
  - Least travel time
  - Waypoints with maximum ratings
  - Waypoints that are open

How it works?

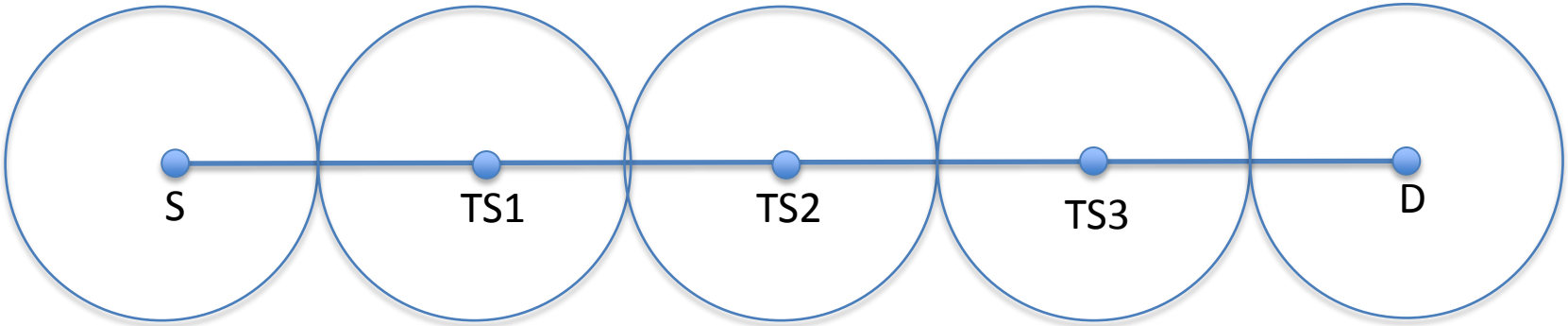
User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

How it works?



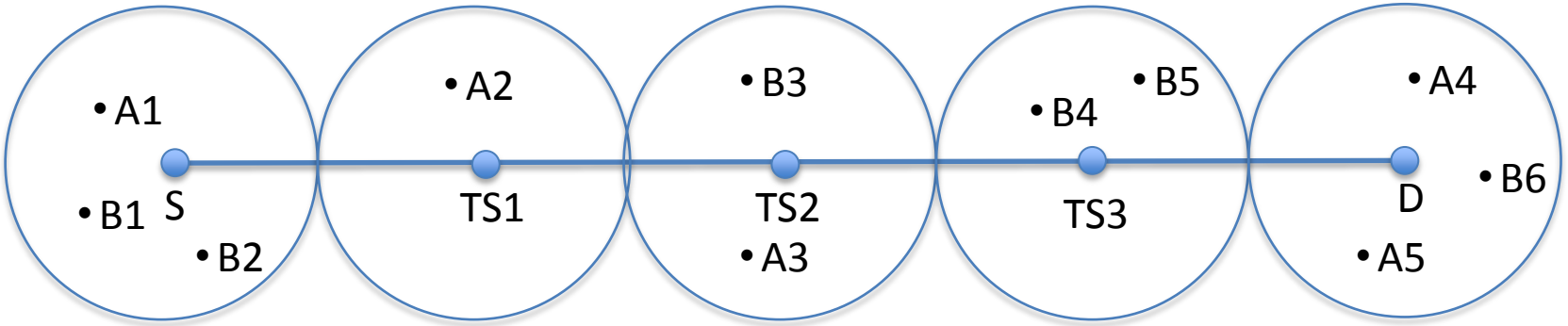
User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

How it works?



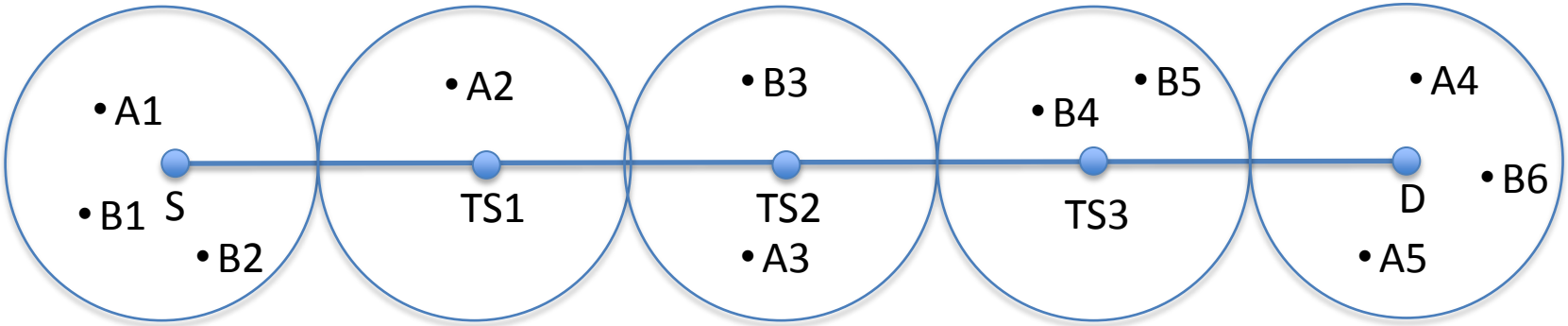
User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

How it works?



User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

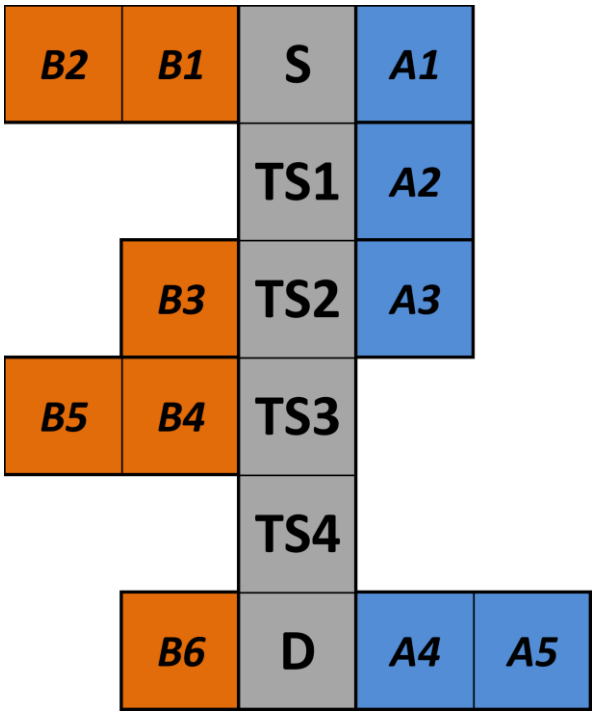
How it works?



Potential POIs are filtered by time, price range, and ratings

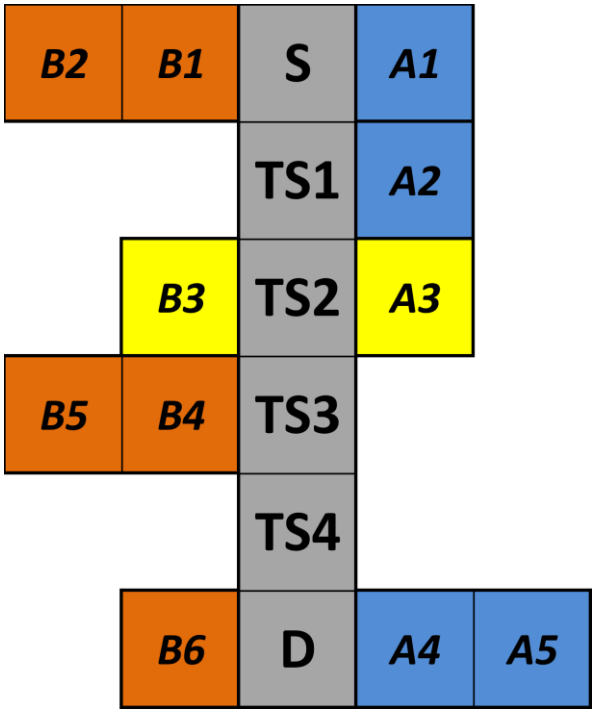
User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

How it works?

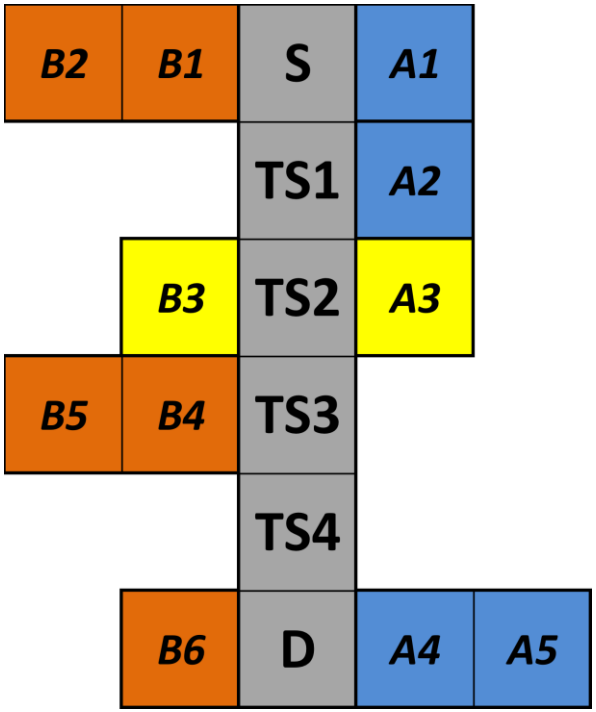


- From S to D, several difference combinations of picking A and B possible
- Pick one with least COST.

How it works?

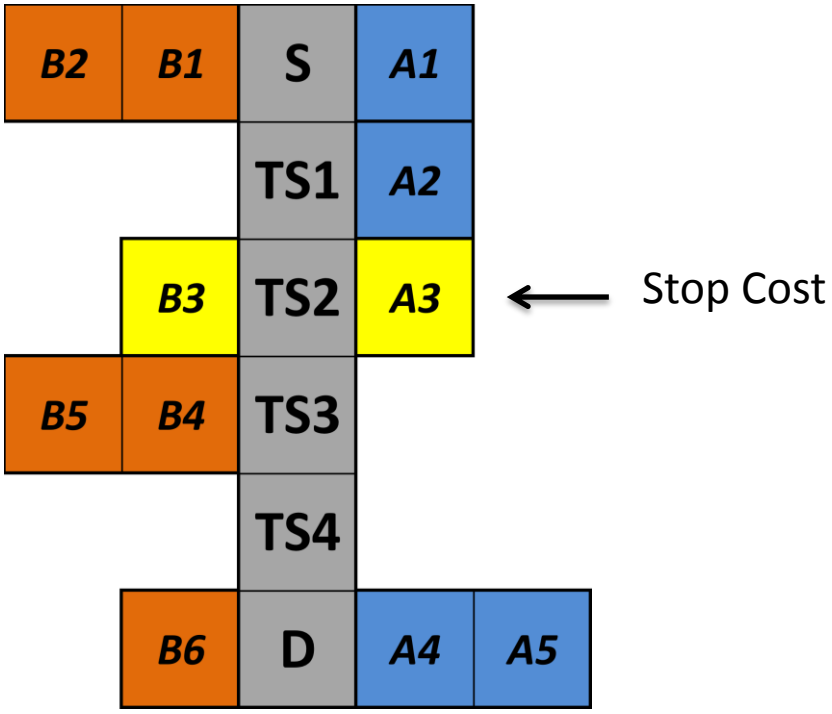


How it works?


B2
B1
S
A1
TS1
A2
B3
TS2
A3
B5
B4
TS3
TS4
B6
D
A4
A5

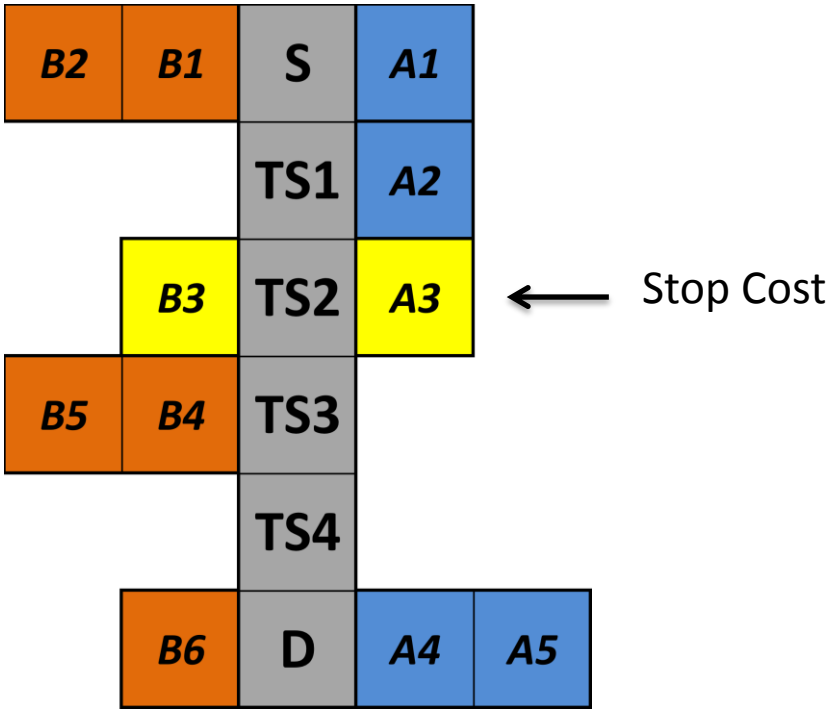

TC
WC
WC
WC
TC
S
TS2
A3
B3
TS2
D

How it works?


B2
B1
S
A1
TS1
A2
B3
TS2
A3
Stop Cost
B5
B4
TS3
TS4
B6
D
A4
A5

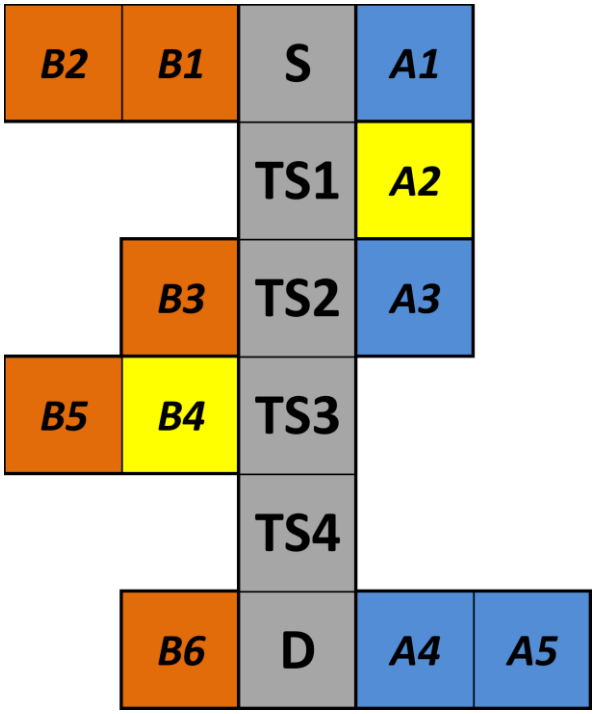

TC
WC
WC
WC
TC
S
TS2
A3
B3
TS2
D

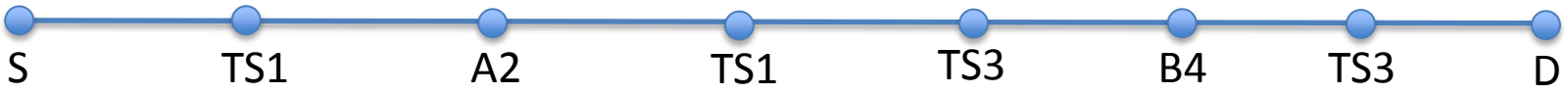
How it works?



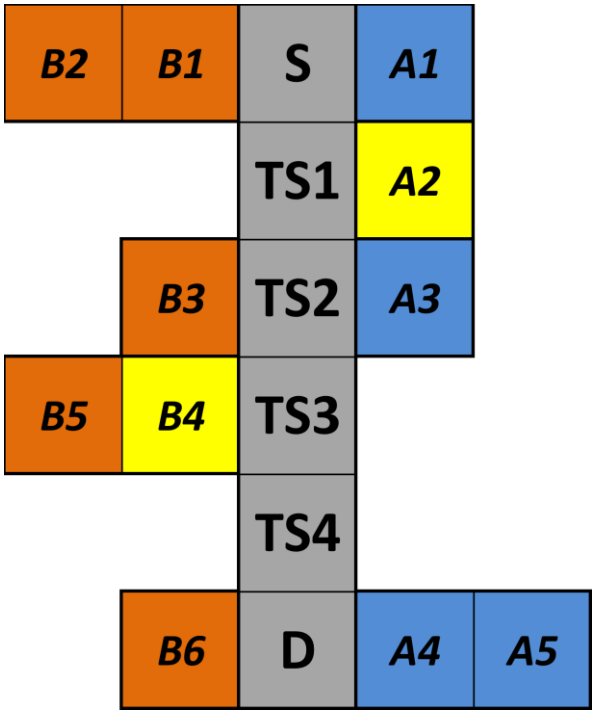
Total Cost = [Dist(S, TS2) + Dist(TS2, D)]*transit_factor +
[Dist(TS2, A3) + Dist(A3, B3) + Dist(B3, TS2)]*walking_factor +
Stop_cost – [Rating(A3) + Rating(B3)]*Rating_factor

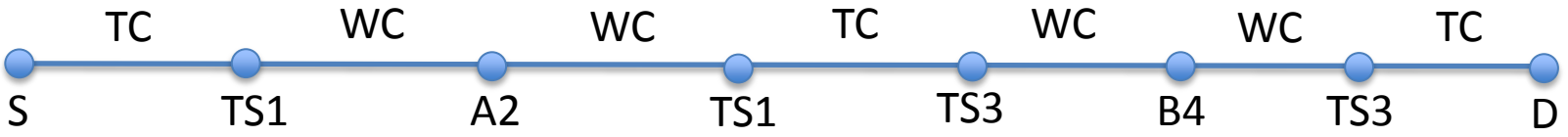
How it works?


B2
B1
S
A1
TS1
A2
B3
TS2
A3
B5
B4
TS3
TS4
B6
D
A4
A5

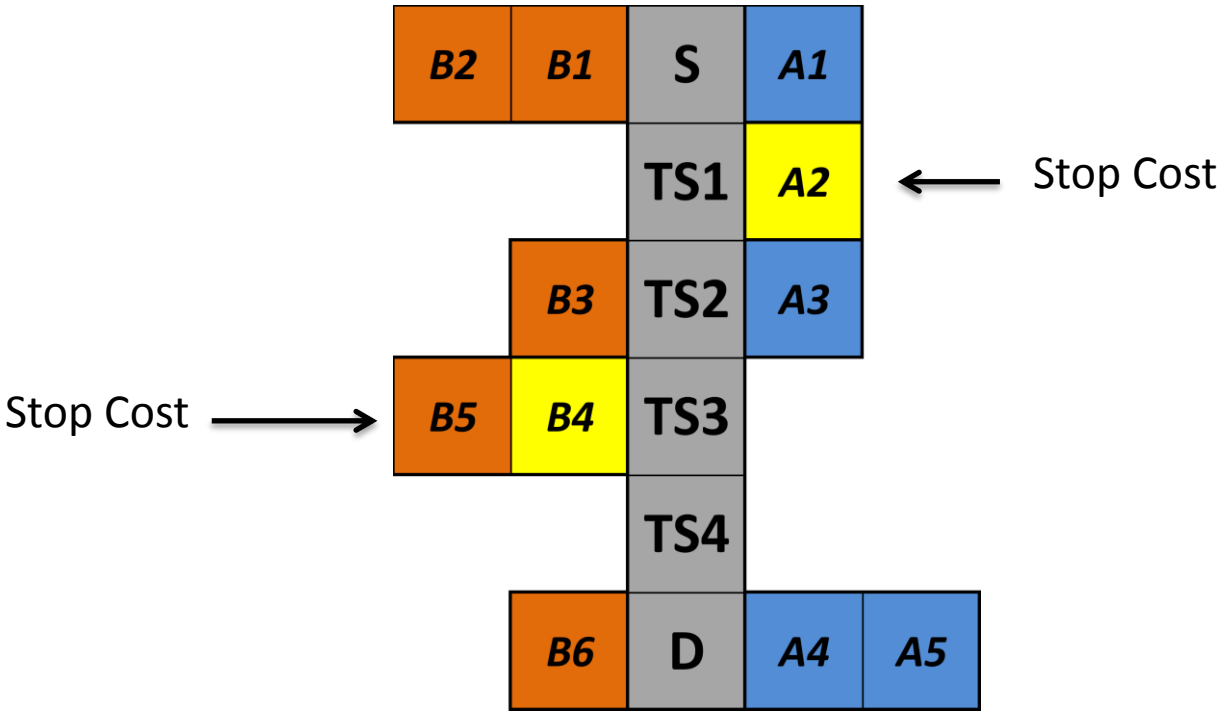

S
TS1
A2
TS1
TS3
B4
TS3
D

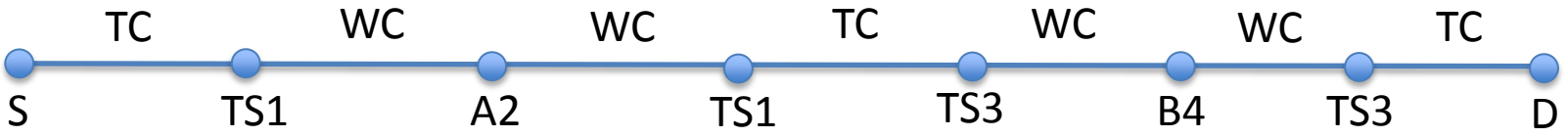
How it works?


B2
B1
S
A1
TS1
A2
B3
TS2
A3
B5
B4
TS3
TS4
B6
D
A4
A5

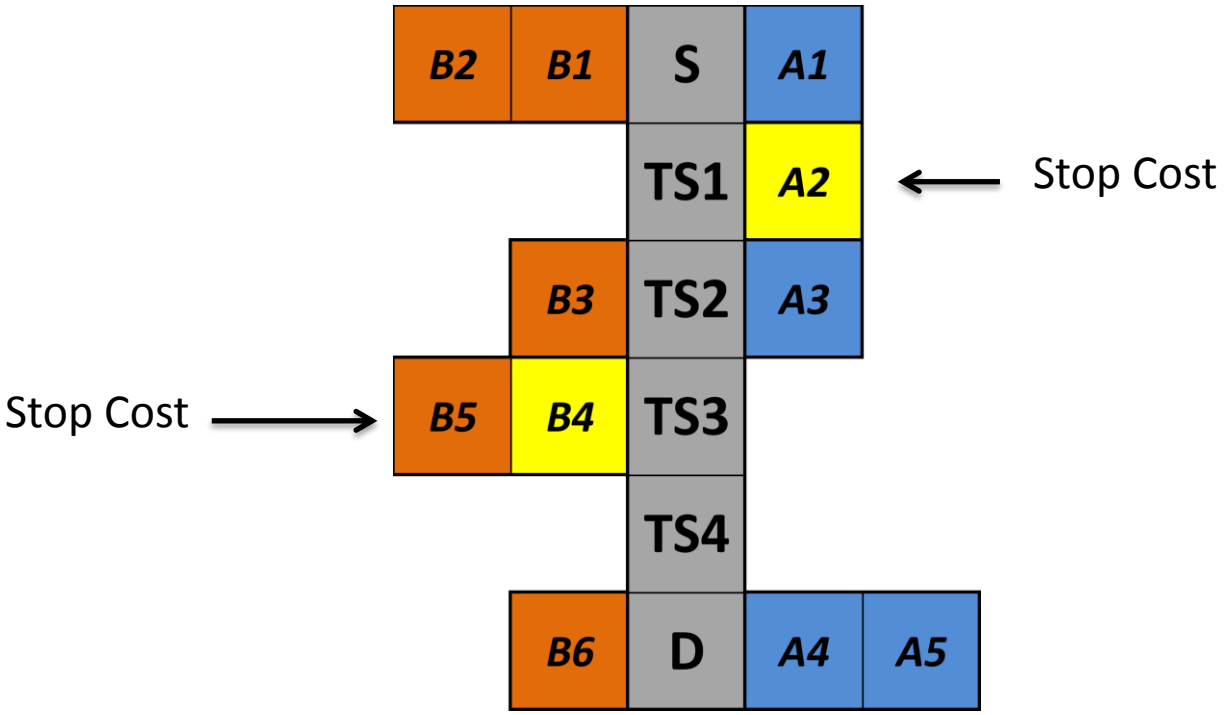

TC
WC
WC
TC
WC
WC
TC
S
TS1
A2
TS1
TS3
B4
TS3
D

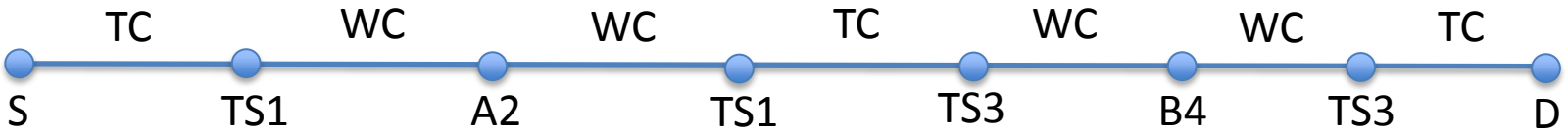
How it works?


B2
B1
S
A1
TS1
A2
Stop Cost
B3
TS2
A3
Stop Cost
B5
B4
TS3
TS4
B6
D
A4
A5


TC
WC
WC
TC
WC
WC
TC
S
TS1
A2
TS1
TS3
B4
TS3
D

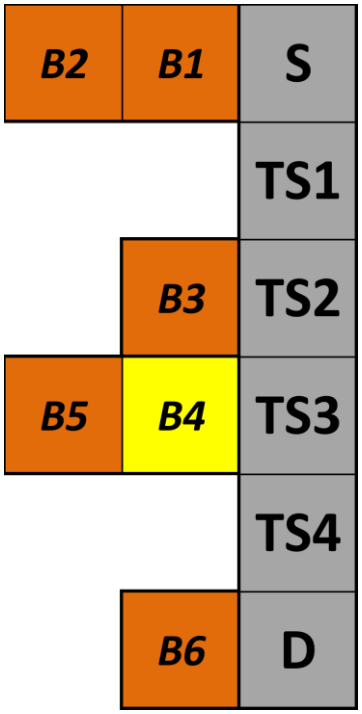
How it works?



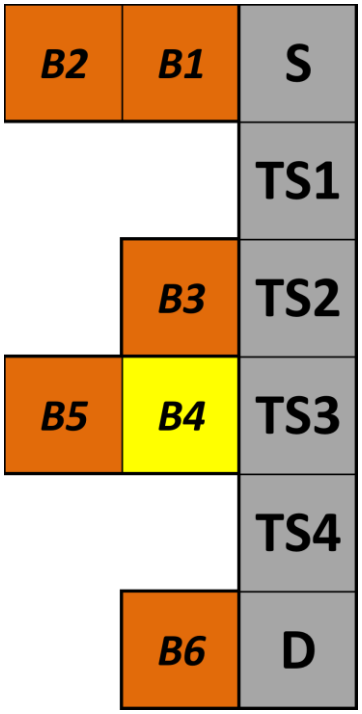


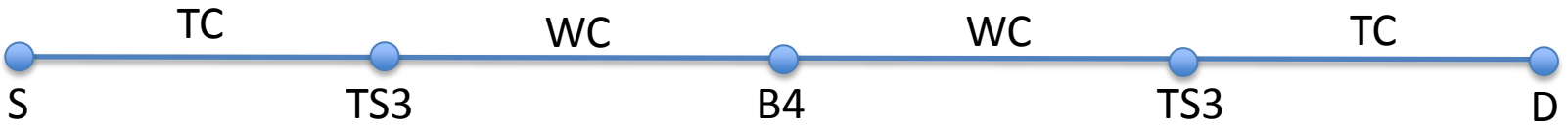
Total Cost = [Dist(S, TS1) + Dist(TS1, TS3) + Dist(TS3, D)]*transit_factor +
2*[Dist(TS1, A2) + Dist(TS3, B4)]*walking_factor +
2*Stop_cost – [Rating(A3) + Rating(B3)]*Rating_factor

How it works?

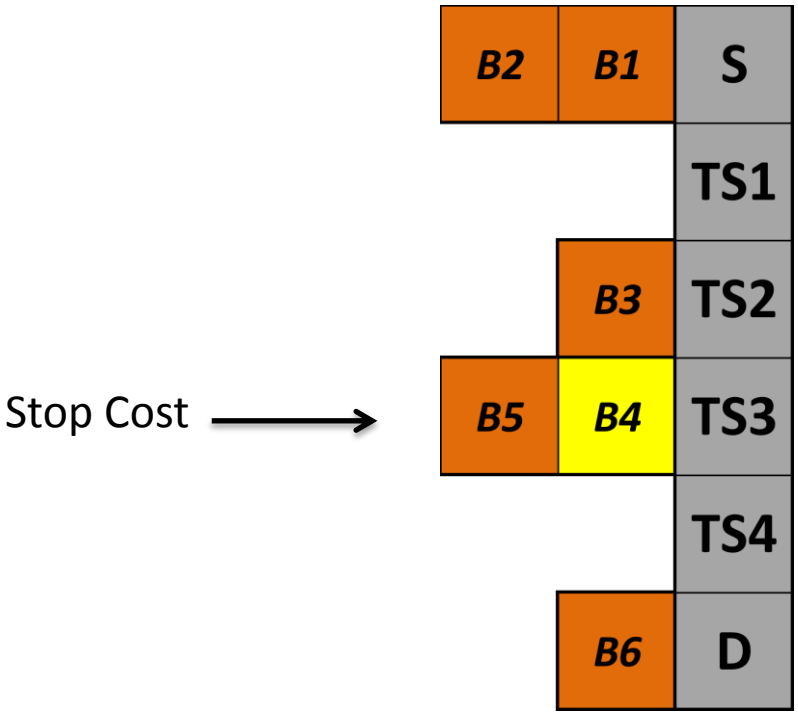


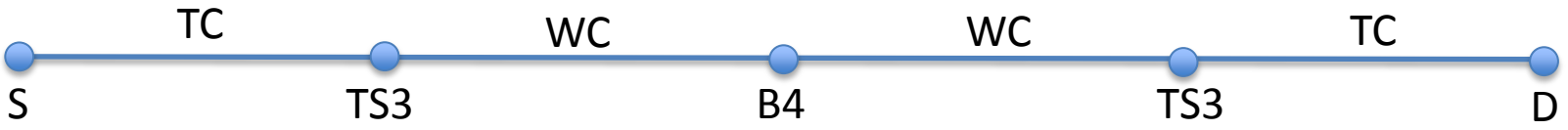
How it works?



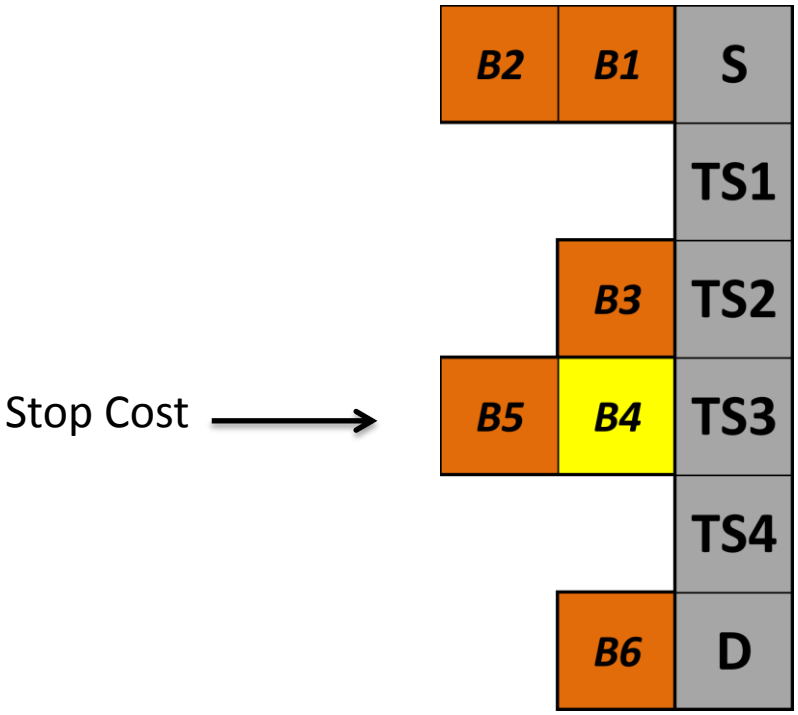


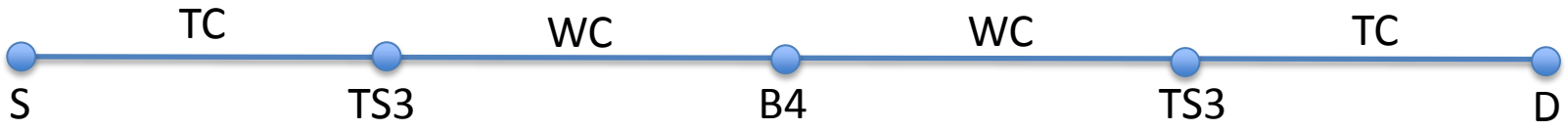
How it works?


B2
B1
S
TS1
B3
TS2
Stop Cost
B5
B4
TS3
TS4
B6
D


TC
WC
WC
TC
S
TS3
B4
TS3
D

How it works?



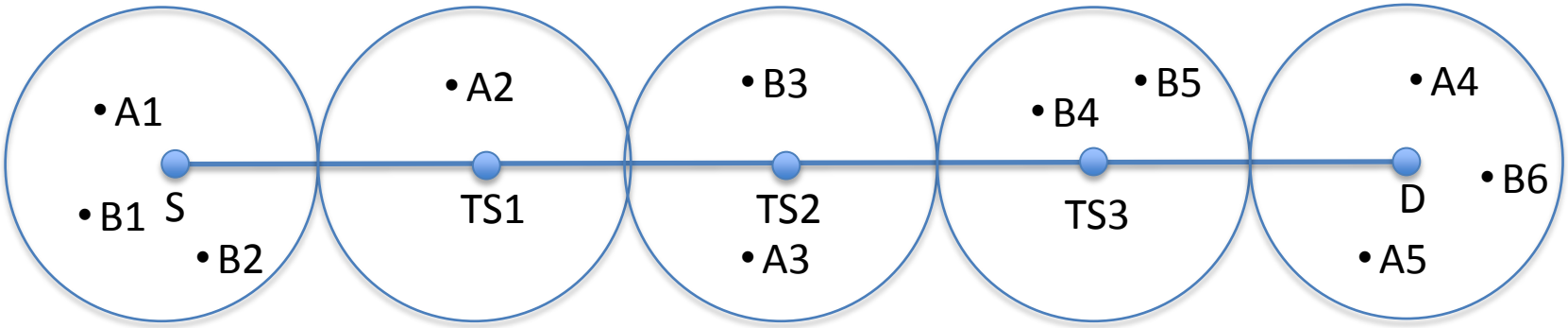


Total Cost = [Dist(S, TS3) + Dist(TS3, D)]*transit_factor +
2*[Dist(TS1, B4)]*walking_factor +
Stop_cost – [Rating(B4)]*Rating_factor

How it works?

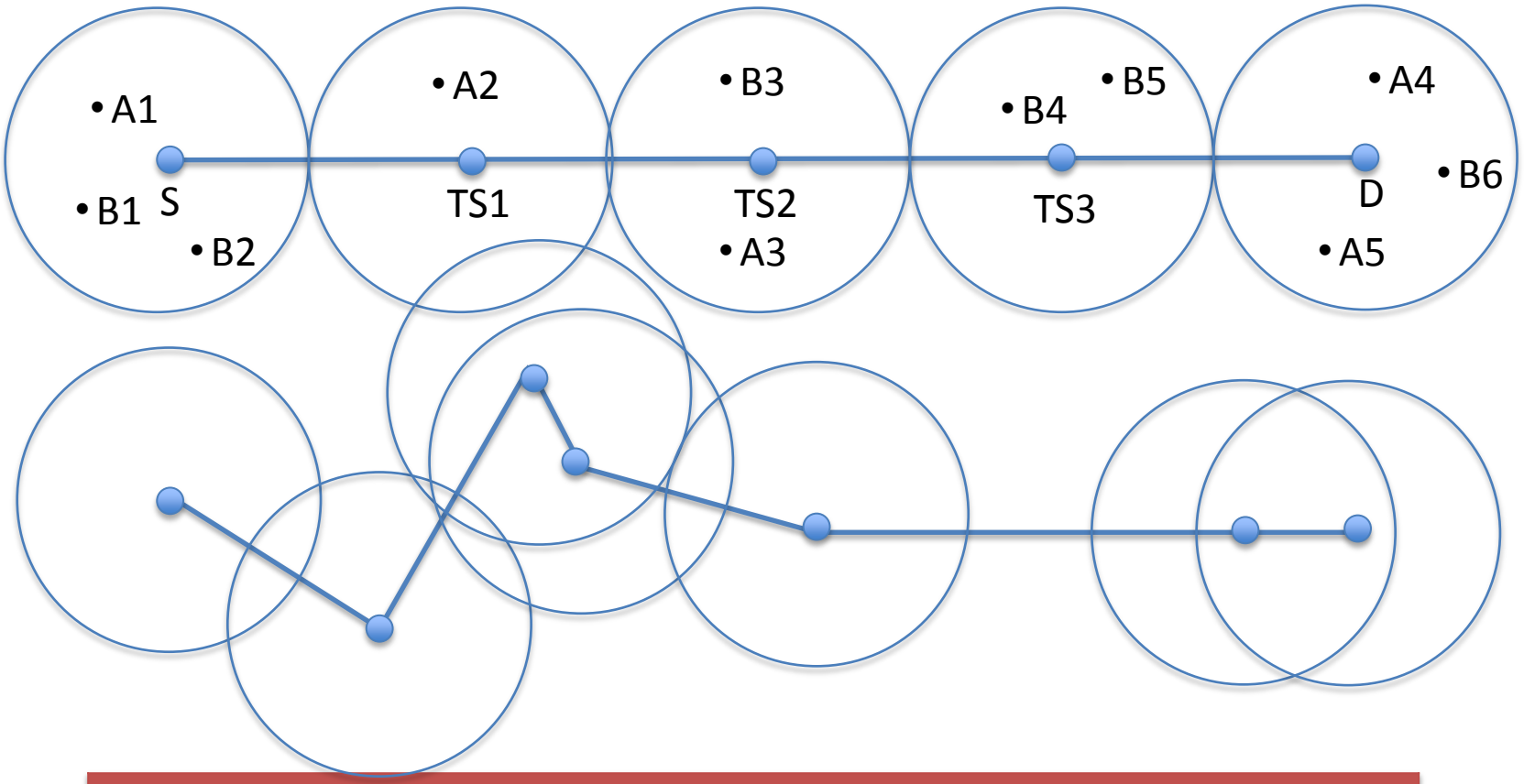
- Identify the top 5 routes with least cost
- For these routes, use Google Direction API to calculate actual route and time
- Show the one with least time

Optimizations



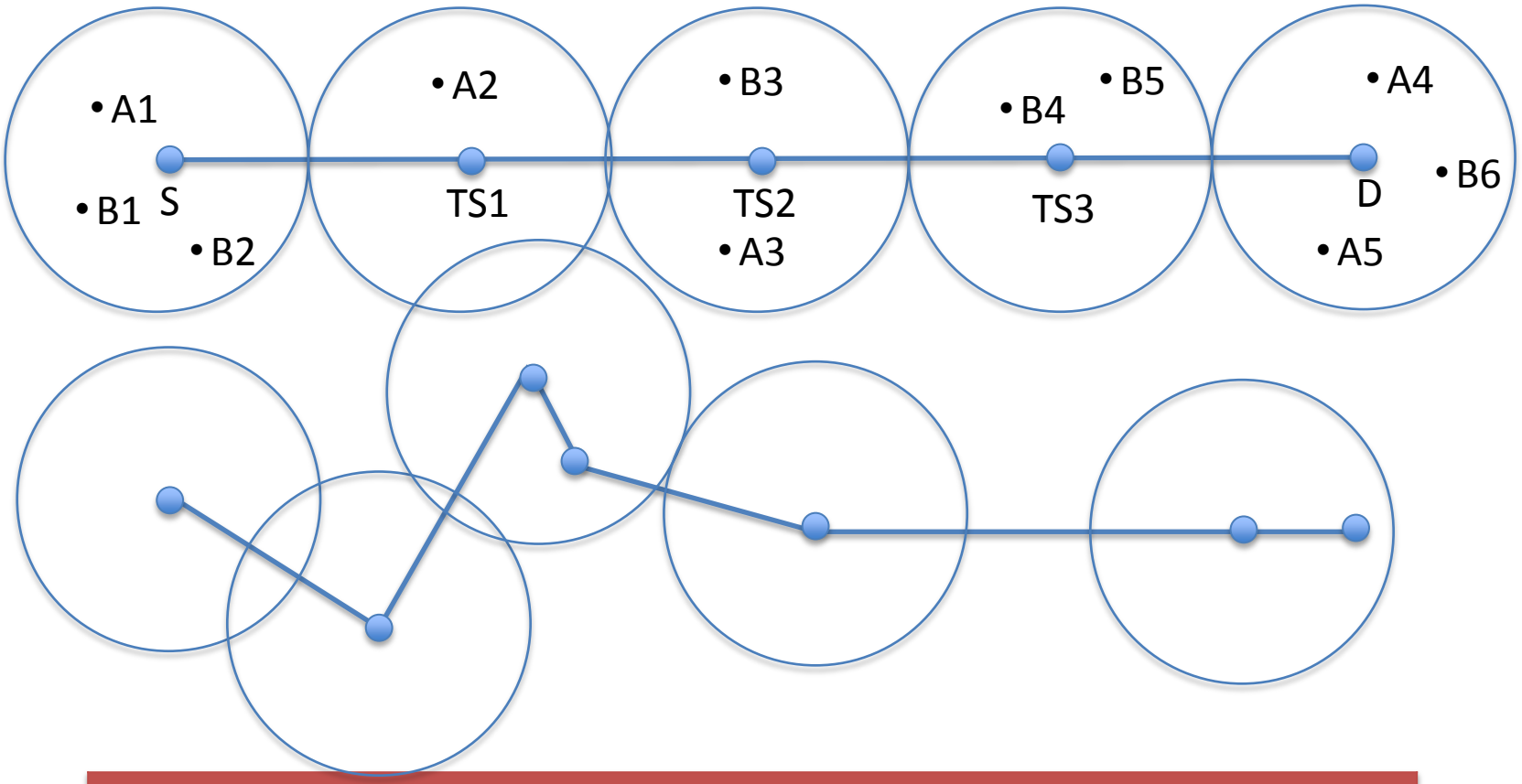
User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

Optimizations



User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

Optimizations



User Inputs	
• Source	• Waypoint A (Starbucks)
• Destination	• Waypoint B (Supermarket)

Architecture

- Client-Server Architecture
- Leverage IaaS from Digital Ocean

Design

- Google APIs:
  - Google Places
  - Google Maps
  - Google Direction
- WMATA API
- Python Web Server
  - Connections with APIs
  - Handles “least costly POI” algorithm
- Android Front End
  - Minimal logic – most computation left for the server

Future Work

- Add more options for users to filter waypoints based on price range and ratings
- Support more than 2 waypoints
- Return multiple routes as suggestions

Questions
?