CS 231

A Network Flow Approach to Bike Sharing

Fatih Bakir, Gabriel Siqueira, Goksu Guvendiren, Mert Kosan, Richika Sharan, Zexi Huang

Outline

1. The Problem

2. Network Flow Approach
   - Basic Model
   - Reductions

3. Experiments & Results

4. Conclusions

The Problem

Bike Sharing

- Bike Sharing systems allow their users to pick up a bike at a station, use it and return it to the same or different station.

- ***Rebalancing Problem:***
    - Different stations can have different levels of supply and demand during the day.
    - The natural flow of bikes might not be enough to satisfy the demand.
        - Need to move bikes between stations to meet demand $\longrightarrow$ **Cost**

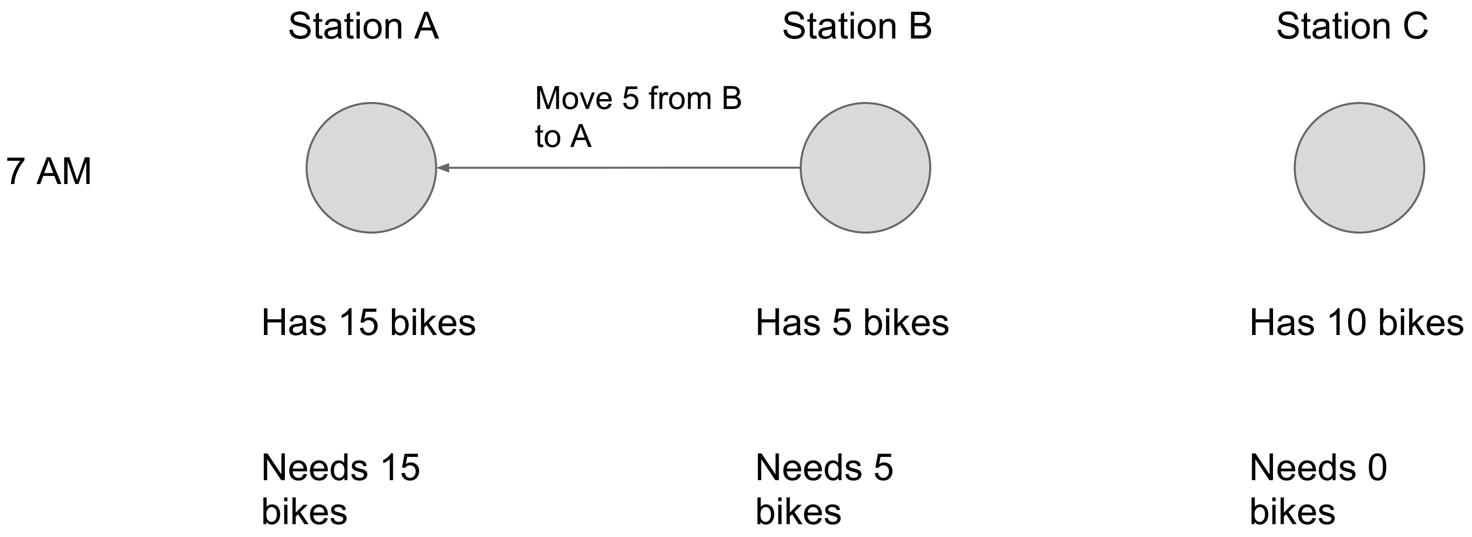
Rebalancing Problem

- Can we minimize our moving costs and still keep everybody happy?

	Station A	Station B	Station C
7 AM			
	Has 10 bikes	Has 10 bikes	Has 10 bikes
	Needs 15 bikes	Needs 5 bikes	Needs 0 bikes

Rebalancing Problem

- Can we minimize our moving costs and still keep everybody happy?



Rebalancing Problem

- Can we minimize our moving costs and still keep everybody happy?

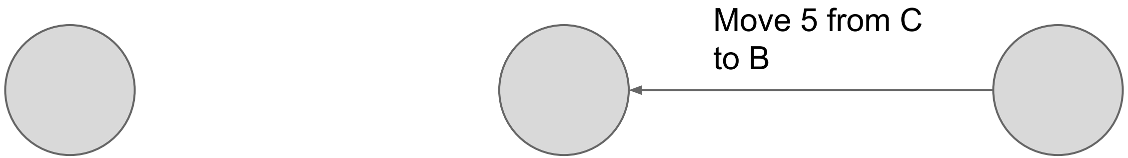
	Station A	Station B	Station C
9 AM			
	Has 0 bikes	Has 0 bikes	Has 10 bikes
	Needs 0 bikes	Needs 5 bikes	Needs 5 bikes

Rebalancing Problem

- Can we minimize our moving costs and still keep everybody happy?

Station A | Station B | Station C

9 AM

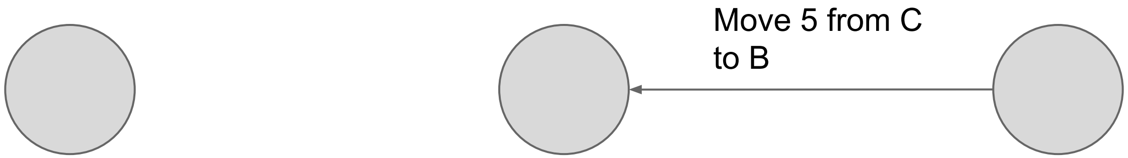


	Station A	Station B	Station C
9 AM	Has 0 bikes	Has 5 bikes	Has 5 bikes
	Needs 0 bikes	Needs 5 bikes	Needs 5 bikes

Rebalancing Problem

- What if we had moved from C to A at 7 AM instead of B to A?
    - This 5 unit deficit in B at 9AM wouldn't have happened. Less money wasted in transfer.

	Station A	Station B	Station C
9 AM			
	Has 0 bikes	Has 5 bikes	Has 5 bikes
	Needs 0 bikes	Needs 5 bikes	Needs 5 bikes



Assumptions

- The number of bikes in the system remains constant.
  - *Bikes don't break, aren't stolen, etc.*

- The total demand at any time slot is always smaller than the total number of bikes.

- The cost only comes from rebalancing bikes.

- The total demand across bike stations at time slot *t* will be equal to the total supply across bike stations at time slot *t+1* .
  - Bikes picked up at time *t* will always be dropped-off at time *t+1*.

- Bikes are only rebalanced 2 times in a day.

Notations

- n fixed-location bike stations
- m total bikes in the system.
- K total number of contiguous days we want to optimize
- k timeslots used to divide a day. (e.g. k=2, day divided into 12h slots)
- b cost of transporting each bike.

Notations

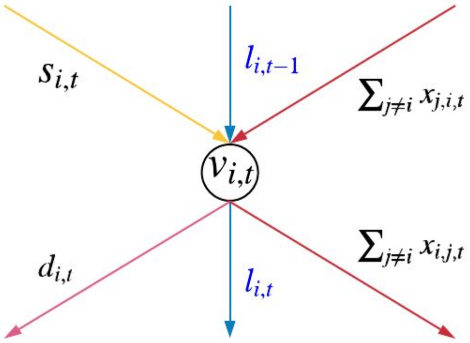
For each time slot t:

- The natural demand of bikes at station i: $d_{i,\,t}$
  - Caused by users picking-up bikes at station i
- The natural supply of bikes at station i: $s_{i,\,t}$
  - Caused by users dropping-off bikes at station i
- Bikes moved by us from bike station i to bike station j: x_{ijt}
- Leftover bikes at bike station i: l_{it}

Bike Dynamics

Bike dynamics follow this (Equation 1):

$$s_{i,t} + l_{i,t-1} + \sum_{j \neq i} x_{j,i,t} = d_{i,t} + l_{i,t} + \sum_{j \neq i} x_{i,j,t}$$



Objective Function

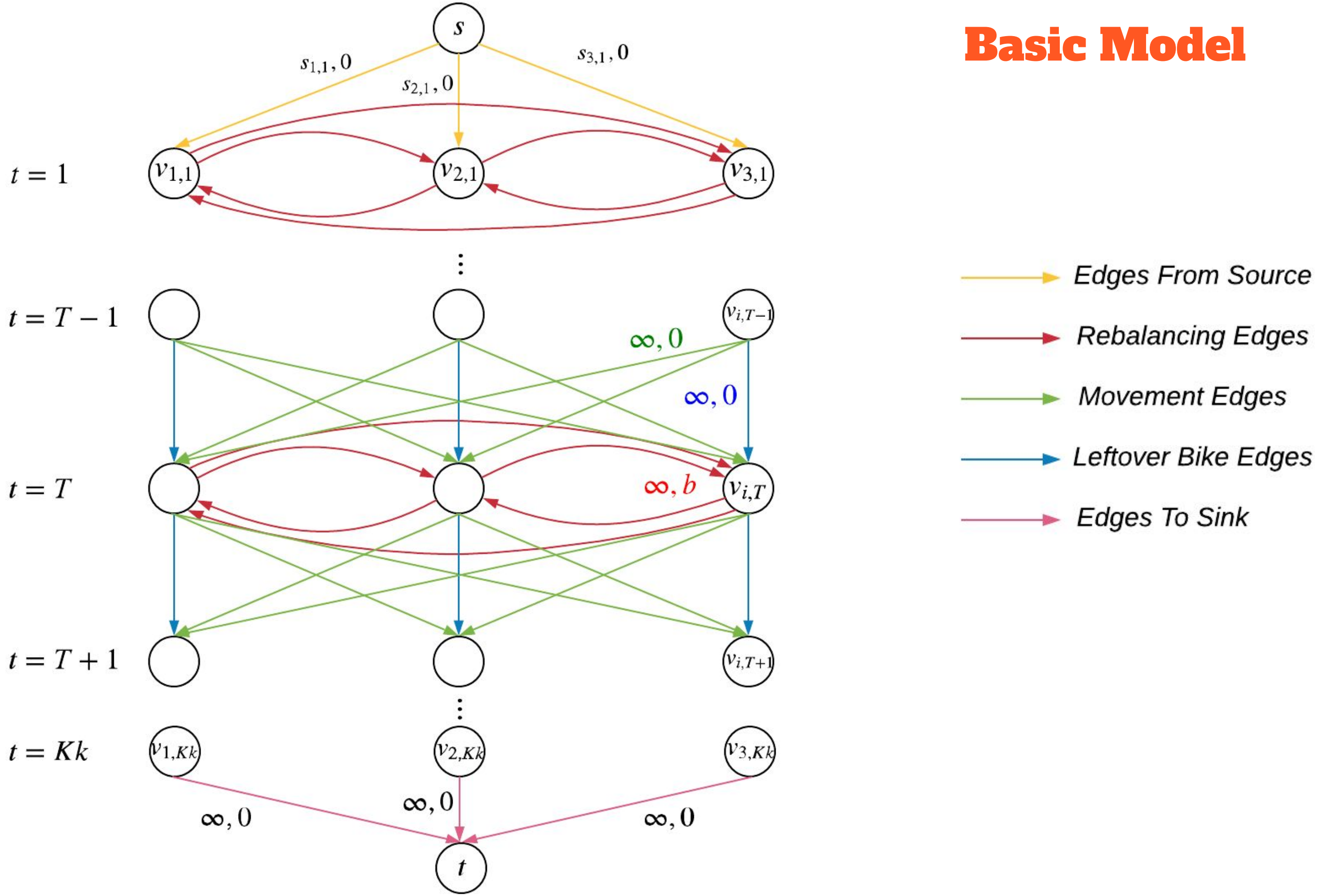
We want to minimize the total cost of our relocation (Equation 2):

$$\min \sum_{i \neq j} \sum_{t=1}^{Kk} b x_{i,j,t}$$

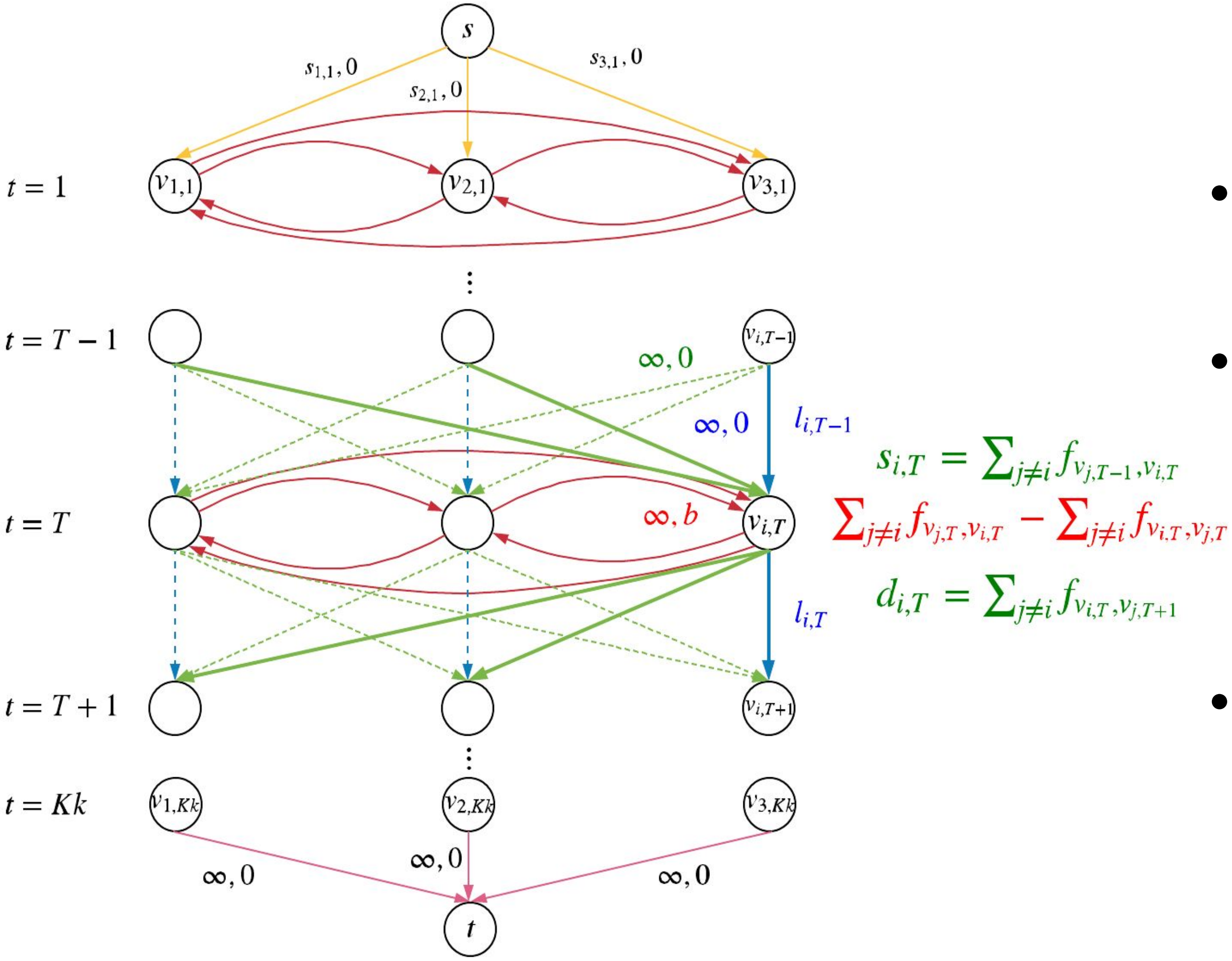
Network Flow Approach

Basic Model

Basic Model



s
$s_{1,1}, 0$
$s_{2,1}, 0$
$s_{3,1}, 0$
$t = 1$
$v_{1,1}$
$v_{2,1}$
$v_{3,1}$
$t = T - 1$
$v_{i,T-1}$
$\infty, 0$
$\infty, 0$
$t = T$
∞, b
$v_{i,T}$
$t = T + 1$
$v_{i,T+1}$
$t = Kk$
$v_{1,Kk}$
$v_{2,Kk}$
$v_{3,Kk}$
$\infty, 0$
$\infty, 0$
$\infty, 0$
t
Edges From Source
Rebalancing Edges
Movement Edges
Leftover Bike Edges
Edges To Sink



Proof of Correctness

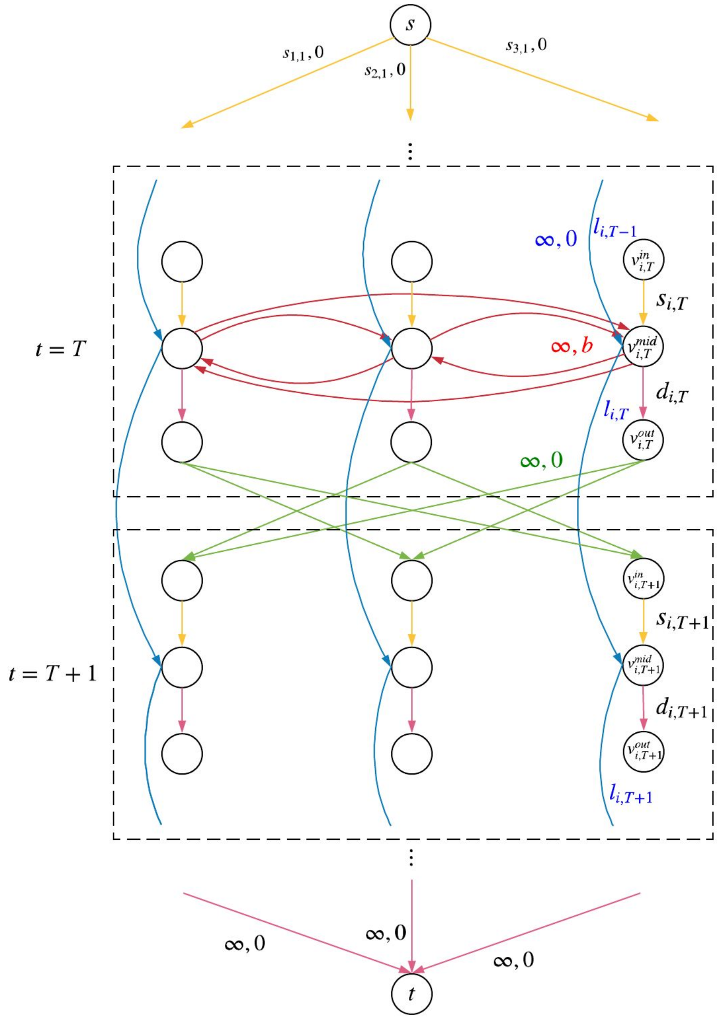
- Supplies and demands are captured by flow constraints on incoming edges and outgoing edges.
- Bike dynamics (Equation 1) is captured by flow conservation at every node:

$$l_{i,T} + d_{i,T} + \sum_{j \neq i} f_{v_{i,T}, v_{j,T}} =$$
$$l_{i,T-1} + s_{i,T} + \sum_{j \neq i} f_{v_{j,T}, v_{i,T}}$$

- Cost of flow of this model is the same as our objective function (Equation 2):

$$L = \sum_{i \neq j} \sum_{t=1}^{Kk} b f_{v_{i,t}, v_{j,t}}$$

Three Node Model



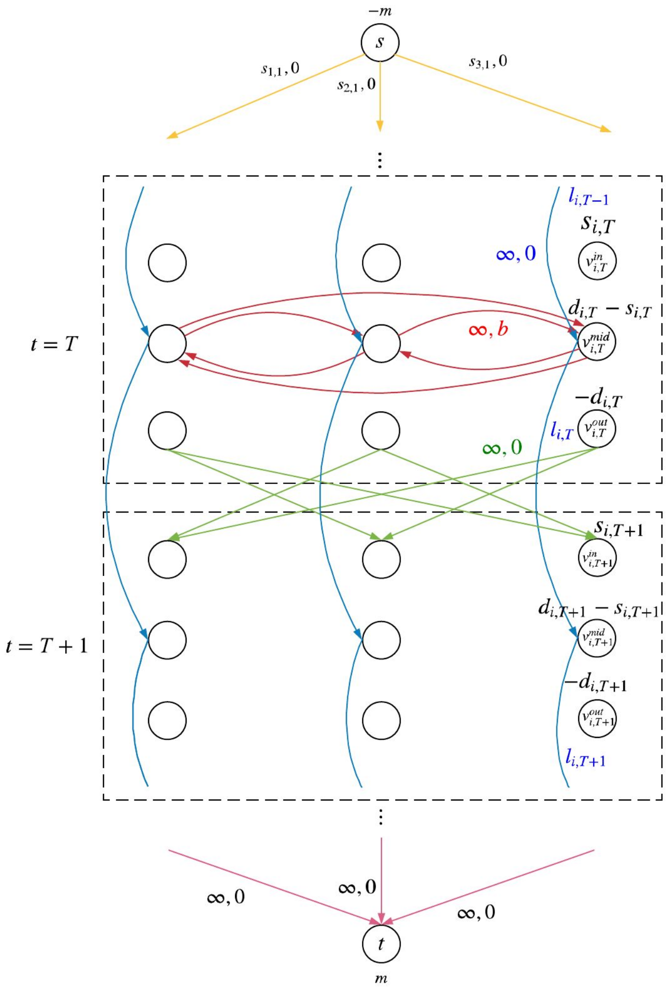
Three Node Model

- We split each node into three nodes: $v_{i,t}^{in}$ $v_{i,t}^{mid}$ and $v_{i,t}^{out}$

Proof of Correctness

- Supplies and demands are captured by flow constraint on $(v^{in}_{i,t}, v^{mid}_{i,t})$ and $(v^{mid}_{i,t}, v^{out}_{i,t})$.
- Bike dynamics (Equation 1) is captured by flow conservation at $v_{i,t}^{mid}$.
- Cost of flow of this model is still the same as our objective function (Equation 2).

Reduced Model



Reduced Model

Supply Edges

Rebalancing Edges

Movement Edges

Leftover Bike Edges

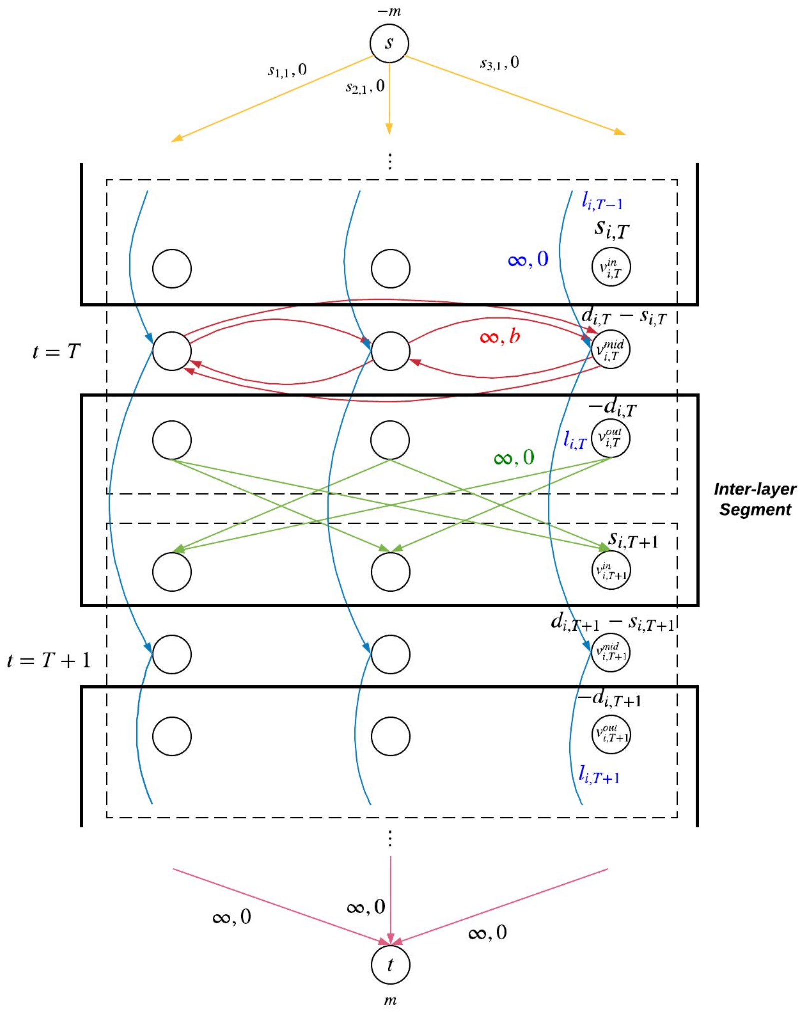
Demand Edges

Proof of Correctness

- Incorporate lower bound on edges by changing to demand/supply of nodes.
- Incorporate exact flow values of demand and supply by removing edges $(v_{i,t}^{in}, v_{i,t}^{mid})$ and $(v_{i,t}^{mid}, v_{i,t}^{out})$.
- Cost of flow of this model is still the same as our objective function (Equation 2).

Final Model

Final Model



Supply Edges

Rebalancing Edges

Movement Edges

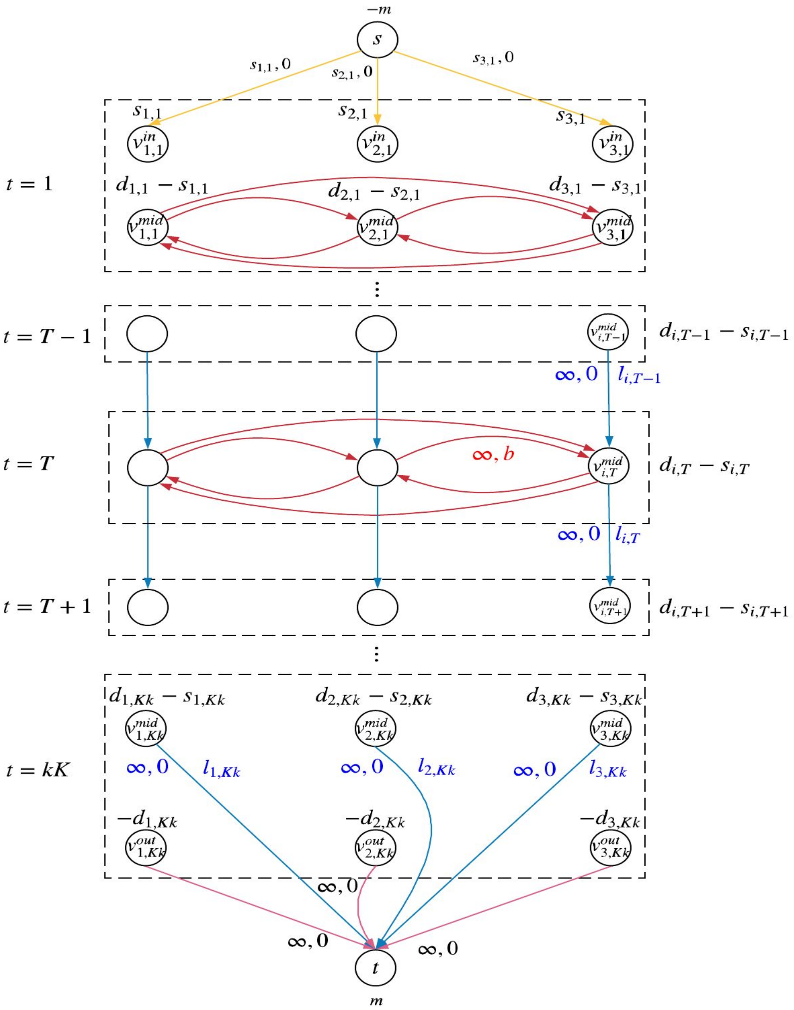
Leftover Bike Edges

Demand Edges

We remove all the nodes in the inter-layer segments and edges between them.

Inter-layer segment forms a closed system as:

- $\sum_i d_{i,t-1} = \sum_i s_{i,t}$
- Net flow across this segment to any other segment will be zero.
- Net cost will also be zero as all edges in this segment have zero cost.



Final Model

Supply Edges

Rebalancing Edges

Leftover Bike Edges

Demand Edges

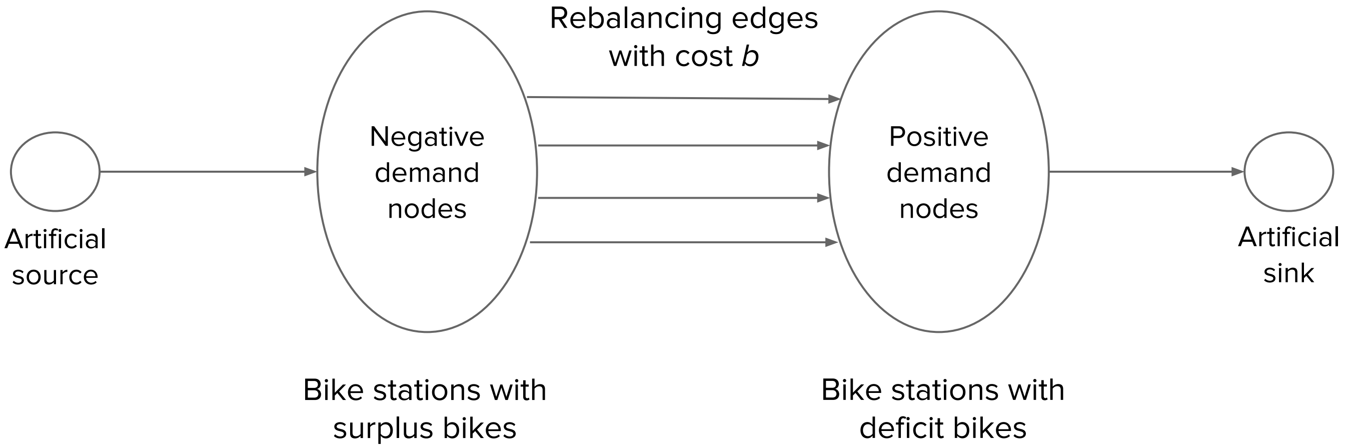
Proof of Correctness

We want to minimize our cost function:

$$Cost = \sum_{i \neq j} \sum_{t=1}^{Kk} b f_{v^{mid}_{i,t}, v^{mid}_{j,t}}$$

which is the same as minimizing our objective function (Equation 2).

Where the cost comes from?



Mincost Solver and Complexity

- In our final model, we have three types of nodes:
  - Source and sink: s and t
  - Nodes in the first supply layer and last demand layer: $v_{i,1}^{in}$ and $v_{i,Kk}^{out}$
  - Nodes in the middle: $v_{i,t}^{mid}$
- Total number of nodes: $|V|=2+2n+nKk$
- And three types of edges:
  - Edges for the first supply layer and last demand layer
  - Leftover edges
  - Rebalancing edges
- Total number of edges: $|E|=2n+nKk+2Kn(n-1)$
- Mincost solver [1]: $O(|E|^2 \log |V| + |E|\ |V| \log^2 |V|)$

[1]James B Orlin. A faster strongly polynomial minimum cost flow algorithm. Operations research, 41(2):338–350, 199

Experiments & Results

Data Generation

- We experimented exclusively with artificial data to guarantee feasibility.

- A feasible, zero-cost solution is simulated.
  - Define an initial distribution of bikes.
  - Simulate movement between stations in remaining time slots, for total number of time slots and obeying original constraints.

- Cost is added by re-distributing demand at each transfer time.
  - Guaranteed feasible if original distribution is restored.

Implementation

- Edmonds Karp for the Max Flow
  - Has O(|V| * |E|^2) complexity
- Cycle Canceling for the Min Cost
  - Bellman-ford to find cycles
  - O(|V| * |E|) for each, up to C times
  - O(C * |V| * |E|) complexity
- Overall complexity is larger than the theoretical
  - O(C * |V| * |E| + |V| * |E|^2)
- Boost Graph library in C++

Surprises

- Largest input did not terminate after 17 hours
  - Has 8000 bikes
  - 100 Nodes
  - 24 time slots, 2 transports a day
  - 30 Days

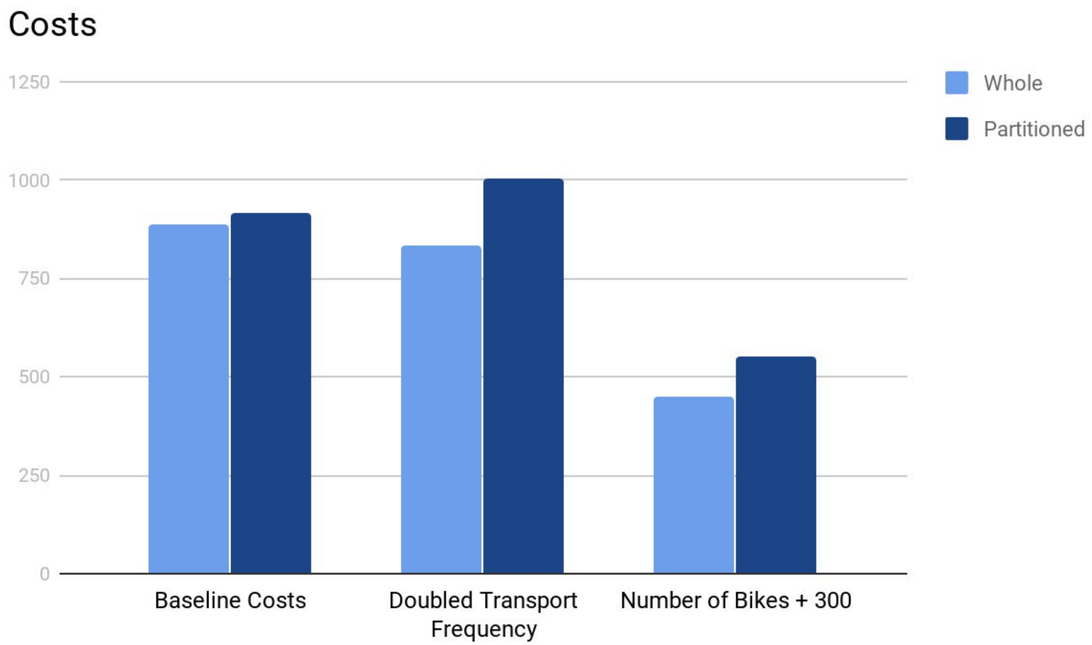
- Partitioning the graph helped us
  - Split the graph
  - Run iteratively
  - Feed the outputs to the next iteration
  - Profit $$

Experiments

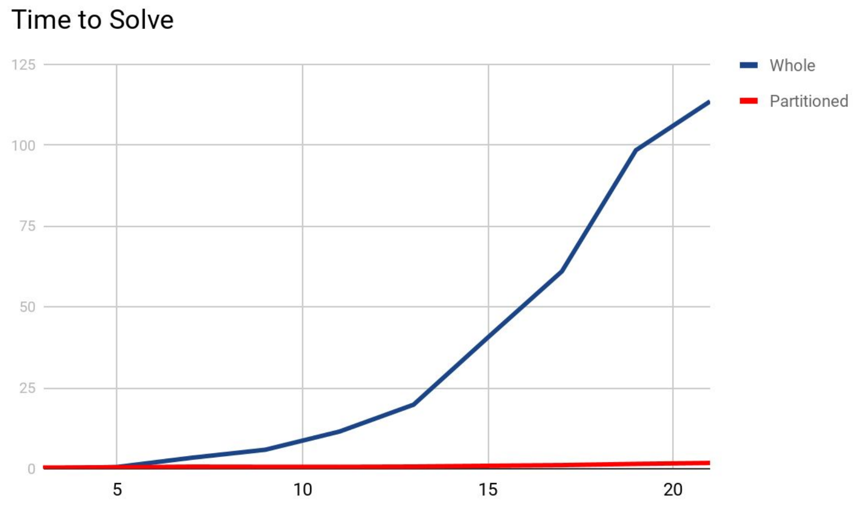
- How does run time increase?
  - Increase the number of stations
  - Increase the number of days

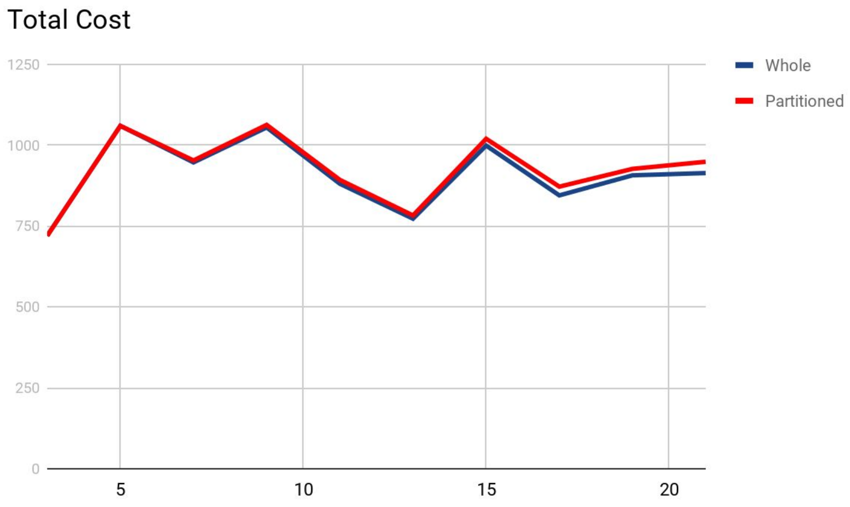
- How does the cost change?
  - Change the frequency of transports
  - Partition the days and solve greedily

Number of Bikes and the Frequency

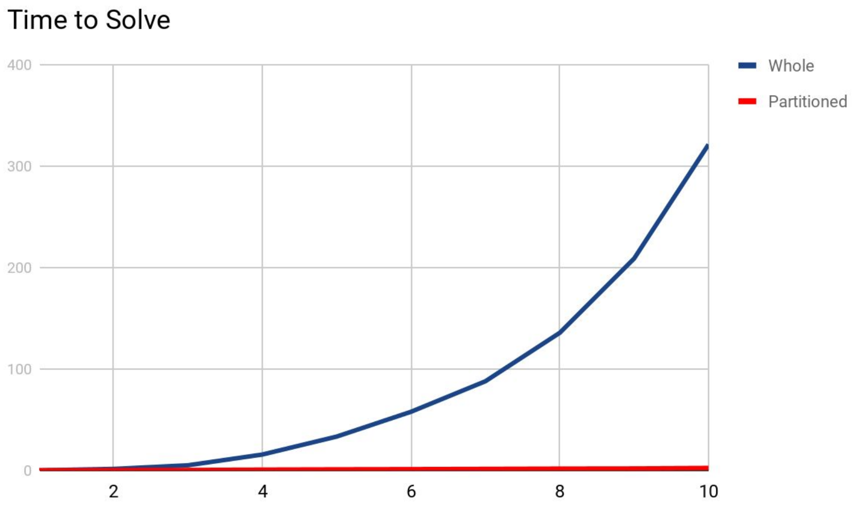


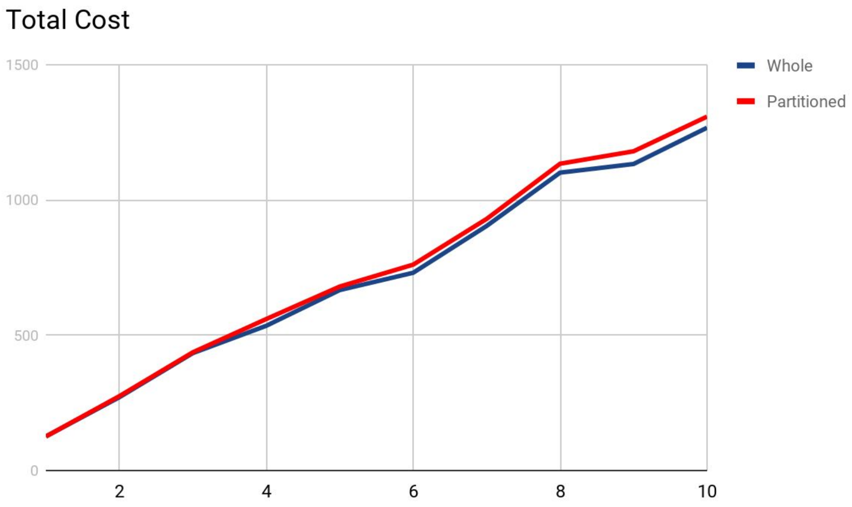
Increasing Bikestops





Increasing Number of Days





Conclusion

Conclusion

- An algorithm that finds the best solution is really expensive.
- Using a greedy approach finds a solution that is :
  - Far more scalable!
  - The increase in the cost is acceptable.

- The greedy algorithm we used is a very naive one.
  - Can be optimized further.