Real Time Routing in Road Networks

Aakriti Gupta

Advisors: Dr. J. Lakshmi, Prof. S. K. Nandy
Cloud Systems Lab, CADL, SERC
Indian Institute of Science

aakriti@cadl.iisc.ernet.in

June 19, 2014

Introduction

Renewed Interest in Routing Problem for Road Networks

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

- Road junctions as graph vertices

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

- Road junctions as graph vertices
- Connecting road segments as edges

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

- Road junctions as graph vertices
- Connecting road segments as edges
- Static graph: constant edge weight

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

- Road junctions as graph vertices
- Connecting road segments as edges
- Static graph: constant edge weight
- Time Dependent graph: edge weight is a function of current traffic, weather conditions etc.

Introduction

Renewed Interest in Routing Problem for Road Networks

- Increasing popularity of Location Based Services
- Real-Time processing requirement (more on this later)

Modeling Routing Problem for Road Networks

- Road junctions as graph vertices
- Connecting road segments as edges
- Static graph: constant edge weight
- Time Dependent graph: edge weight is a function of current traffic, weather conditions etc.
- Now use any shortest path algorithm on this graph, like Dijkstra's search[1].

So what has been done so far?

So what has been done so far?

...A lot!

So what has been done so far?

...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations

So what has been done so far?

...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations
- Fastest known query time, hub labeling[5] computes shortest path on road networks of Europe or the USA in fraction of a microsecond[6]

So what has been done so far?

...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations
- Fastest known query time, hub labeling[5] computes shortest path on road networks of Europe or the USA in fraction of a microsecond[6]

But the problem isn't solved yet!

So what has been done so far?

...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations
- Fastest known query time, hub labeling[5] computes shortest path on road networks of Europe or the USA in fraction of a microsecond[6]

But the problem isn't solved yet!

- Most of these approaches haven't shown success in dynamic case[7]

So what has been done so far?

...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations
- Fastest known query time, hub labeling[5] computes shortest path on road networks of Europe or the USA in fraction of a microsecond[6]

But the problem isn't solved yet!

- Most of these approaches haven't shown success in dynamic case[7]
- These algorithms involve 2 steps - preprocessing (slow) and online computation (fast)

So what has been done so far?

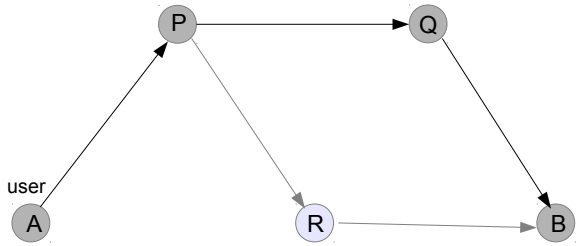
...A lot!

- For static road networks, many heuristics based algorithms[2][3][4] exist to speed up the shortest path computations
- Fastest known query time, hub labeling[5] computes shortest path on road networks of Europe or the USA in fraction of a microsecond[6]

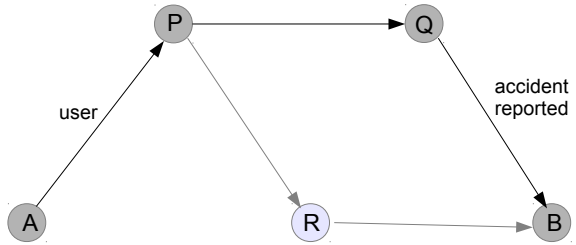
But the problem isn't solved yet!

- Most of these approaches haven't shown success in dynamic case[7]
- These algorithms involve 2 steps - preprocessing (slow) and online computation (fast)
- Unrealistic to do the preprocessing step everytime as graph changes. Periodic updates don't use real time information.

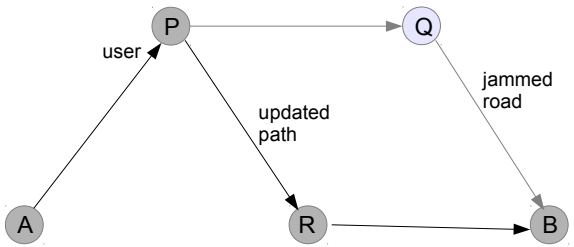
Real-Time processing requirement



(a) t = 0

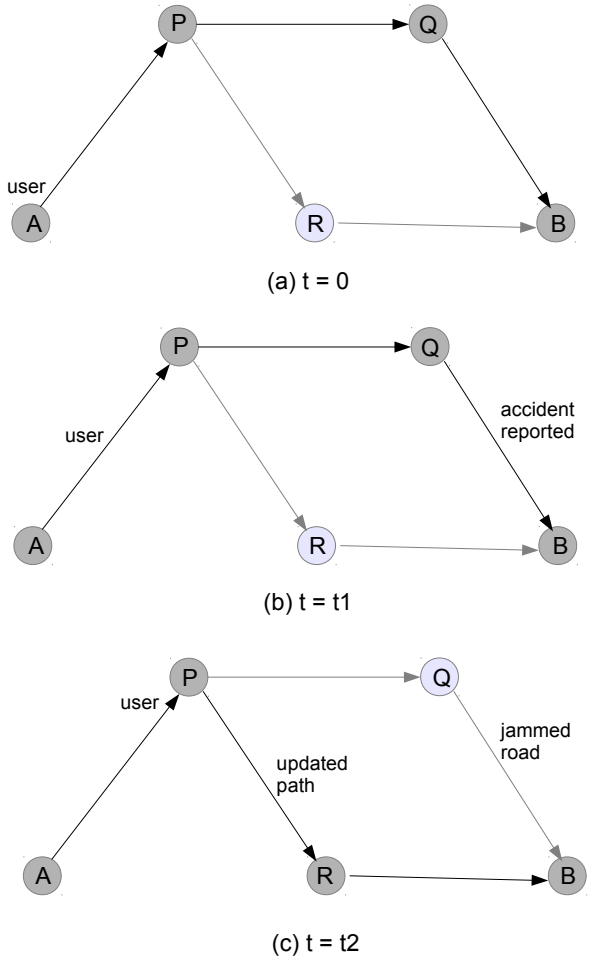


(b) t = t1



(c) t = t2

Real-Time processing requirement



- Real time updates are not fully utilized if revised route not sent to the user
- Need for a proactive system
- Also avoids misdirection! Driver followed satellite navigation instructions in the dark and her car was hit by a train on a rail crossing that was not shown on the system [8].

State of the art for dynamic road networks

State of the art for dynamic road networks

Some interesting approaches

State of the art for dynamic road networks

Some interesting approaches

- Send top k shortest paths to end user and all updates along these paths. User makes the routing decisions [9]

State of the art for dynamic road networks

Some interesting approaches

- Send top k shortest paths to end user and all updates along these paths. User makes the routing decisions [9]
- Game theoretic approach: No central server, cars are the intelligent agents, communicate among each other and compute where to go next [10]

State of the art for dynamic road networks

Some interesting approaches

- Send top k shortest paths to end user and all updates along these paths. User makes the routing decisions [9]
- Game theoretic approach: No central server, cars are the intelligent agents, communicate among each other and compute where to go next [10]

The problem with such decentralized approach is

State of the art for dynamic road networks

Some interesting approaches

- Send top k shortest paths to end user and all updates along these paths. User makes the routing decisions [9]
- Game theoretic approach: No central server, cars are the intelligent agents, communicate among each other and compute where to go next [10]

The problem with such decentralized approach is

- Prohibitive for thin clients: all mobile devices do not have the capability to handle computations and communication that is required.

So far...

So far...

Problems with current frameworks

So far...

Problems with current frameworks

- Clients don't know when to ask for route updates.

So far...

Problems with current frameworks

- Clients don't know when to ask for route updates.
- Current systems model the road traffic based on history and send results based on the time of the day. Real time updates are largely ignored.

So far...

Problems with current frameworks

- Clients don't know when to ask for route updates.
- Current systems model the road traffic based on history and send results based on the time of the day. Real time updates are largely ignored.

Our Approach

So far...

Problems with current frameworks

- Clients don't know when to ask for route updates.
- Current systems model the road traffic based on history and send results based on the time of the day. Real time updates are largely ignored.

Our Approach

- A proactive method for using real-time updates in road networks, can offer better utility for travellers.

So far...

Problems with current frameworks

- Clients don't know when to ask for route updates.
- Current systems model the road traffic based on history and send results based on the time of the day. Real time updates are largely ignored.

Our Approach

- A proactive method for using real-time updates in road networks, can offer better utility for travellers.
- A graph density based method to choose time optimal algorithm for query dependent route computation.

Our approach

Our approach

- Modelling the problem as a real time job scheduling problem.

Our approach

- Modelling the problem as a real time job scheduling problem.
- Job is the (src, dest) pair submitted by remote clients.

Our approach

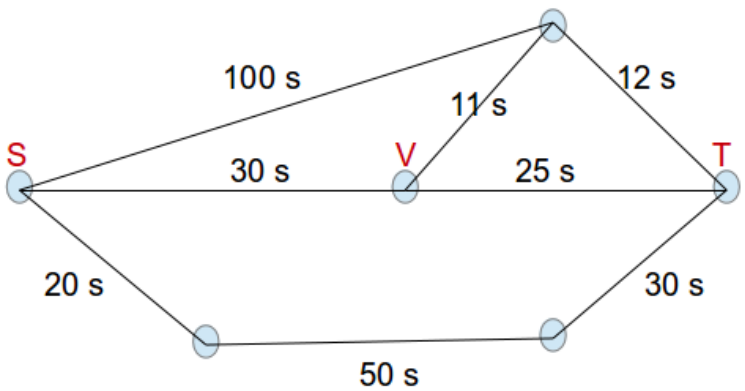
- Modelling the problem as a real time job scheduling problem.
- Job is the (src, dest) pair submitted by remote clients.
- Scheduling is done such that response is computed using real time road network information.

Our approach

- Modelling the problem as a real time job scheduling problem.
- Job is the (src, dest) pair submitted by remote clients.
- Scheduling is done such that response is computed using real time road network information.
- After route is computed for (s,t) pair, with 'v' as next stop, a refresh job (v,t) is added. Result of this refresh job is supposed to reach user before user reaches 'v'.

Our approach

- Modelling the problem as a real time job scheduling problem.
- Job is the (src, dest) pair submitted by remote clients.
- Scheduling is done such that response is computed using real time road network information.
- After route is computed for (s,t) pair, with 'v' as next stop, a refresh job (v,t) is added. Result of this refresh job is supposed to reach user before user reaches 'v'.



Assumptions

Assumptions

- Users can be polled for their current location.

Assumptions

- Users can be polled for their current location.
- Updates (like traffic, weather conditions) on the graph can be analyzed and converted into edge weights [11].

Assumptions

- Users can be polled for their current location.
- Updates (like traffic, weather conditions) on the graph can be analyzed and converted into edge weights [11].
- Edge weight represents the time it would take to travel on that edge at current time.

Types of jobs

Fresh

- A new user connecting to the system.
- Given a unique ID which represents this user.
- Subsequent refresh and redo jobs carry forward this same ID.

Types of jobs

Fresh

- A new user connecting to the system.
- Given a unique ID which represents this user.
- Subsequent refresh and redo jobs carry forward this same ID.

Refresh

- User is following the system suggested path.
- System pro-actively decides whether the user should continue or switch to a different path.

Types of jobs

Fresh

- A new user connecting to the system.
- Given a unique ID which represents this user.
- Subsequent refresh and redo jobs carry forward this same ID.

Refresh

- User is following the system suggested path.
- System pro-actively decides whether the user should continue or switch to a different path.

Redo

- System was late in giving response to the user or user chooses to take a different path.
- Re-computation is required based on current location.

More about our approach

More about Jobs

More about our approach

More about Jobs

- All jobs are aperiodic
- Independent of each other, don't follow any precedence relations among them.
- Non-preemtive, because a job cannot be paused and resumed as network might have seen updates during this time.

More about our approach

More about Jobs

- All jobs are aperiodic
- Independent of each other, don't follow any precedence relations among them.
- Non-preemtive, because a job cannot be paused and resumed as network might have seen updates during this time.

Job scheduling is...

More about our approach

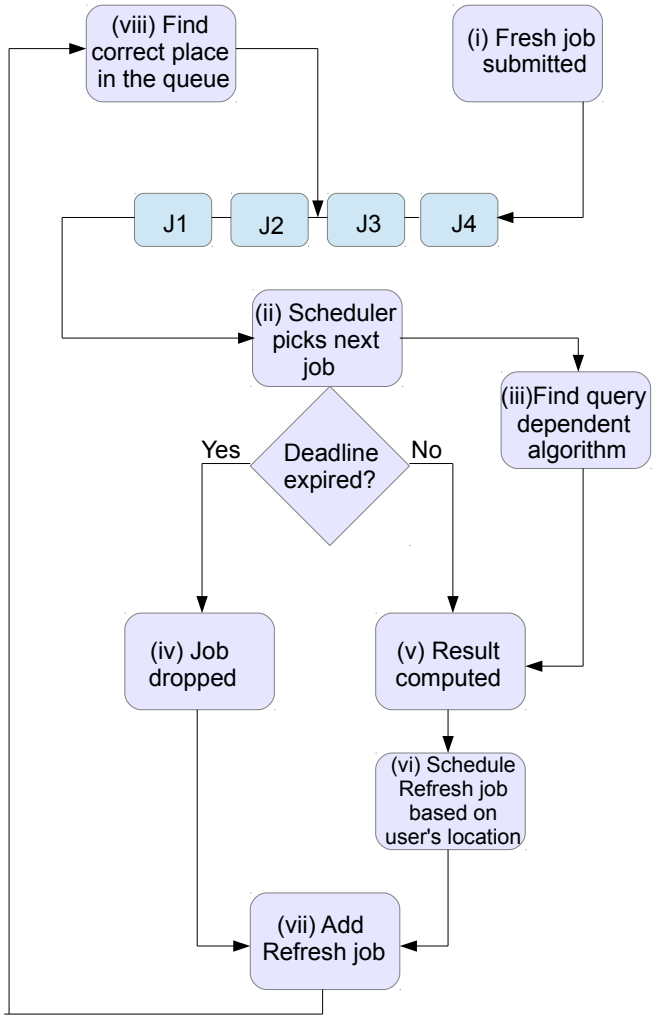
More about Jobs

- All jobs are aperiodic
- Independent of each other, don't follow any precedence relations among them.
- Non-preemtive, because a job cannot be paused and resumed as network might have seen updates during this time.

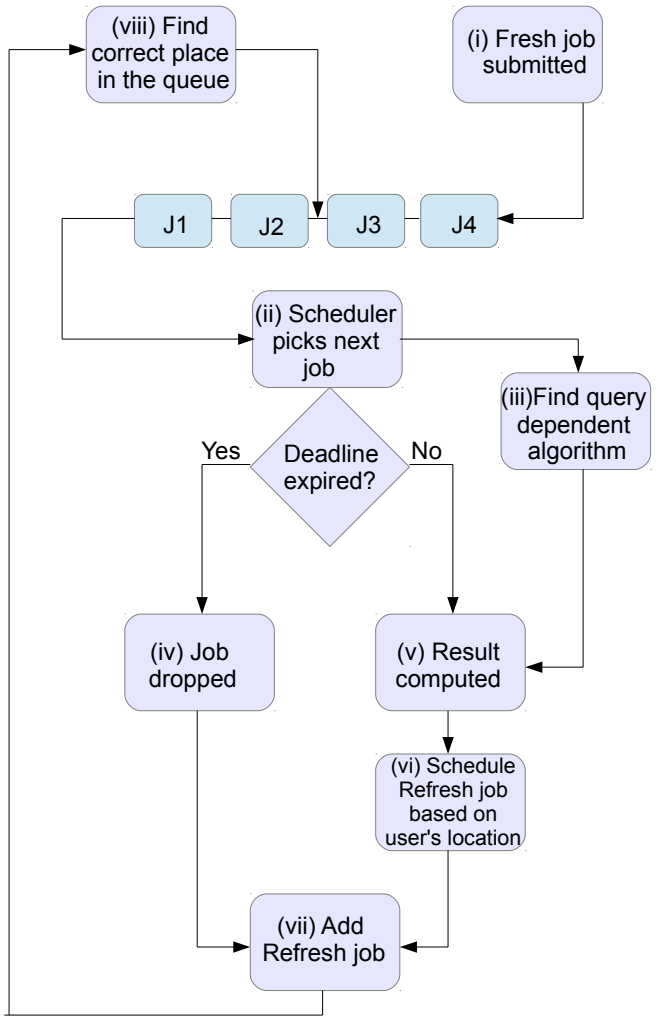
Job scheduling is...

- *Dynamic:* no binding relation between a job and a particular processor.
- *Priority-driven:* scheduling decisions are based on the priorities of the jobs and take place when events such as job completions occur.

Overall System Model

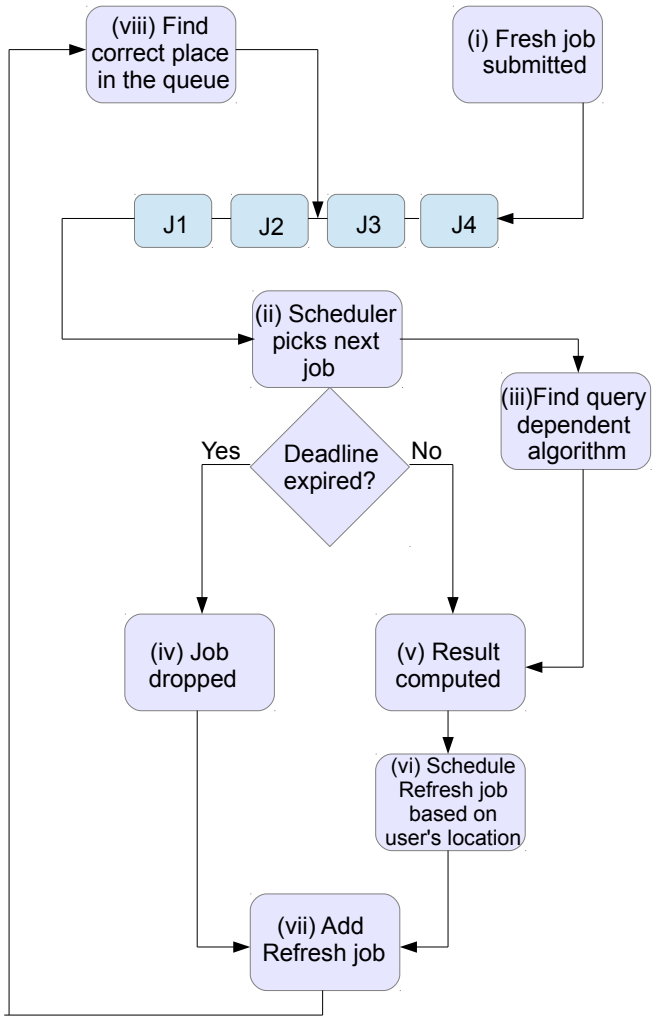


Overall System Model



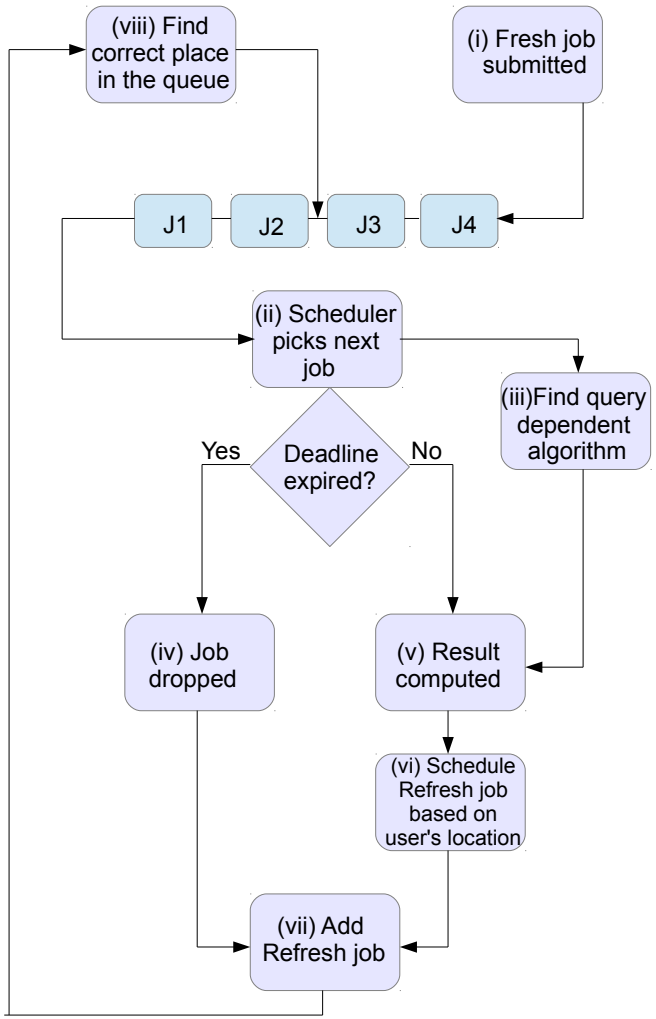
- Fresh jobs enter job queue from one end

Overall System Model



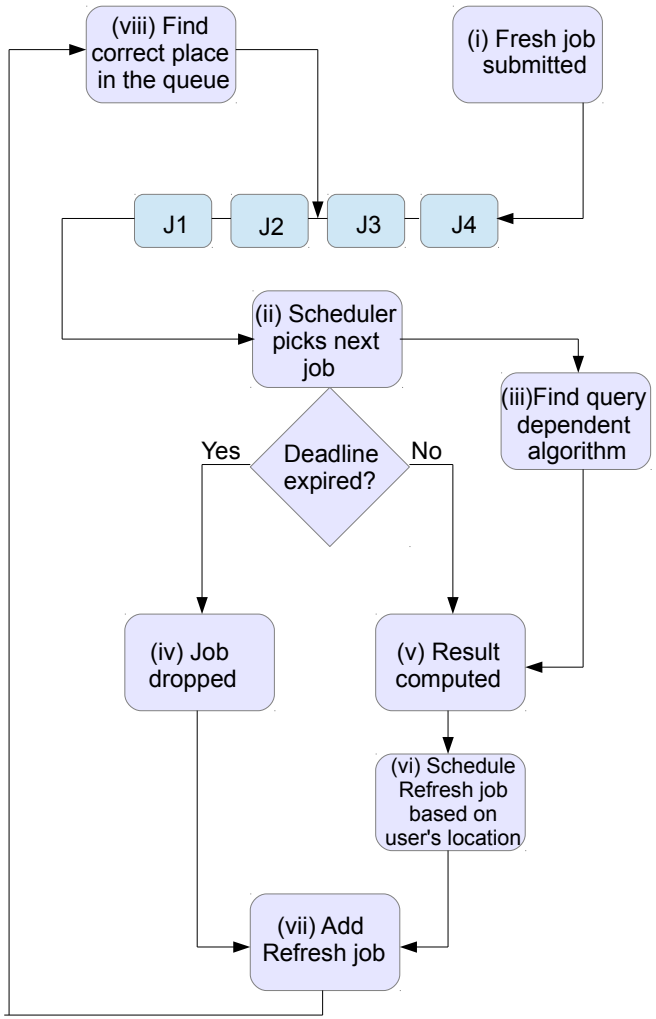
- Fresh jobs enter job queue from one end
- Scheduler finds the next job to be computed.

Overall System Model



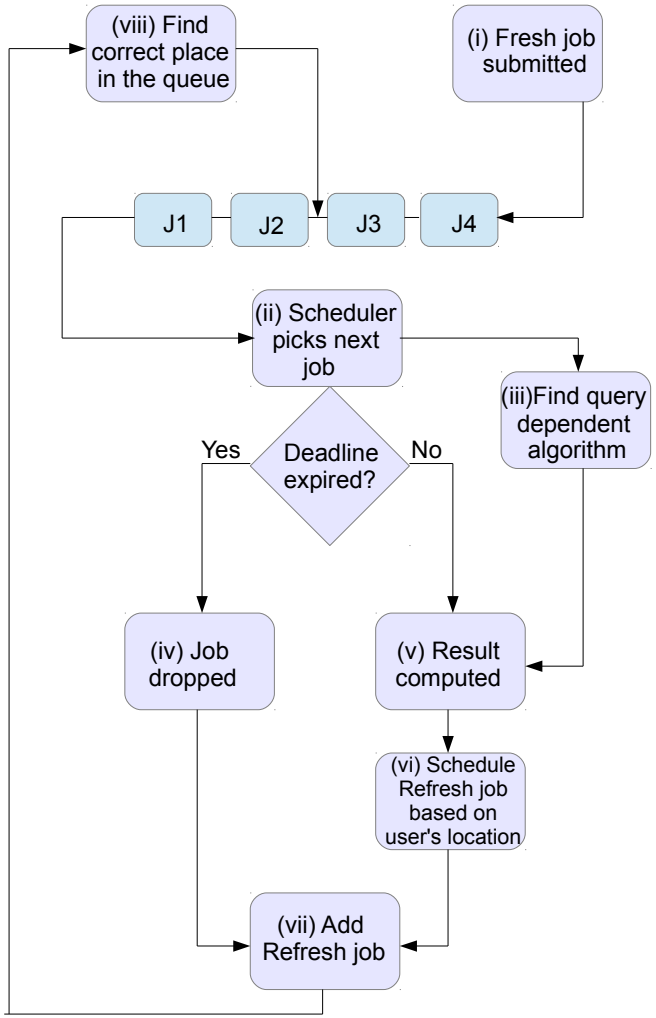
- Fresh jobs enter job queue from one end
- Scheduler finds the next job to be computed.
- If deadline not expired, result is computed. Otherwise dropped.

Overall System Model



- Fresh jobs enter job queue from one end
- Scheduler finds the next job to be computed.
- If deadline not expired, result is computed. Otherwise dropped.
- In either case, refresh job is added.

Overall System Model



- Fresh jobs enter job queue from one end
- Scheduler finds the next job to be computed.
- If deadline not expired, result is computed. Otherwise dropped.
- In either case, refresh job is added.
- Deadline of refresh jobs decides their priority and appropriate place in the job queue.

Computing Job Deadlines

- *fresh* and *redo jobs* are submitted without any deadline and added at the back of the job queue.

Computing Job Deadlines

- *fresh* and *redo jobs* are submitted without any deadline and added at the back of the job queue.
- *refresh job* has 2 timestamps associated with it. It has to be scheduled at any time such that:

Computing Job Deadlines

- *fresh* and *redo jobs* are submitted without any deadline and added at the back of the job queue.
- *refresh job* has 2 timestamps associated with it. It has to be scheduled at any time such that:

$$Timestamp1 \leq Scheduling\_time \leq Timestamp2 \quad (1)$$

Computing Job Deadlines

- *fresh* and *redo jobs* are submitted without any deadline and added at the back of the job queue.
- *refresh job* has 2 timestamps associated with it. It has to be scheduled at any time such that:

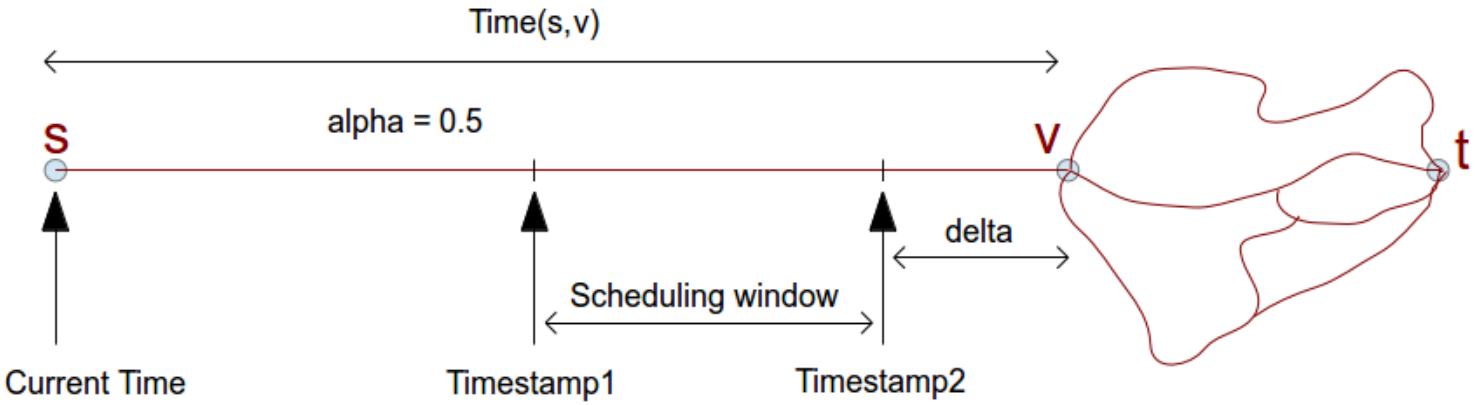
$$Timestamp1 \leq Scheduling\_time \leq Timestamp2 \quad (1)$$

$$Timestamp1 = Current\_time + \alpha * time(s, v) \quad (2)$$

$$Timestamp2 = Current\_time + time(s, v) - \Delta \quad (3)$$

α is any constant ϵ (0,1) and Δ is the estimated upper bound on computation plus communication time.

Illustration

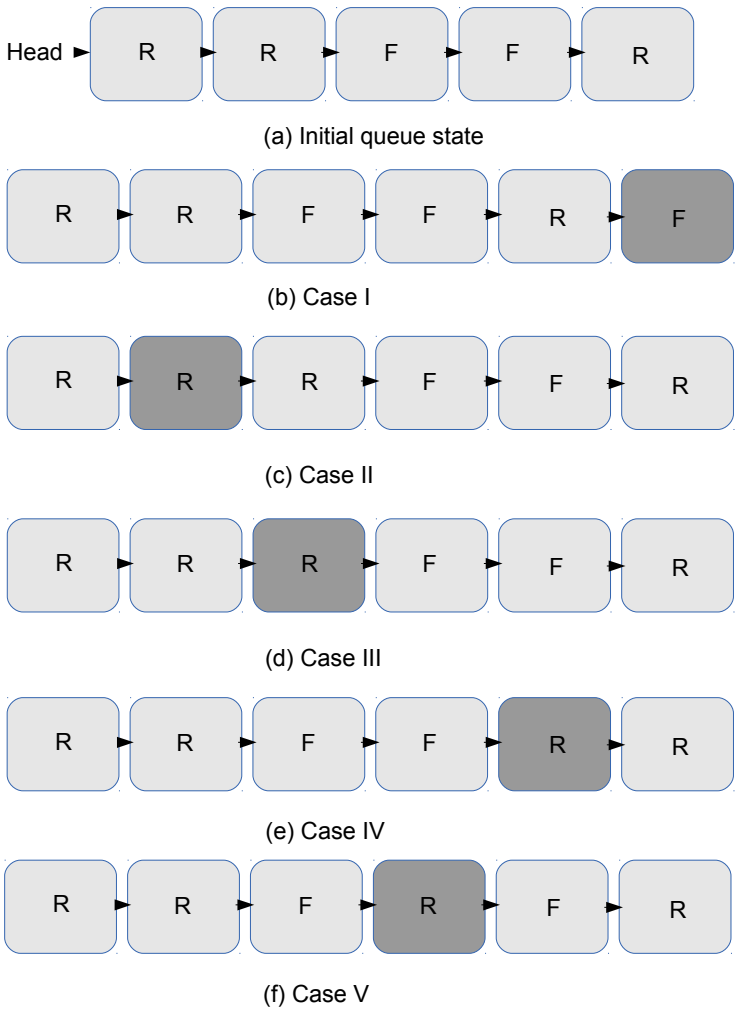


- Route computation before Timestamp1 will not have the lastest information.
- Route computation after Timestamp2 will not be complete before user reaches 'v', hence not useful.

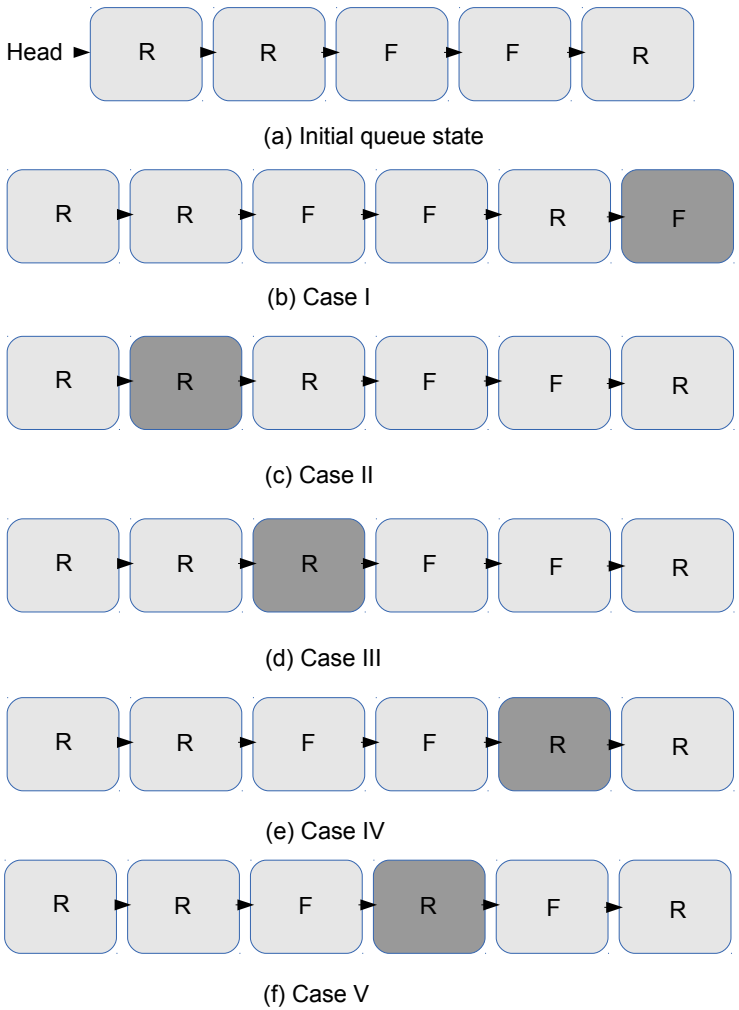
Job Sceduling Algorithm

```
1:  while (1) do
2:      while !(Queue Empty) do
3:          ptr ← queue head
4:          if ptr.jobtype = fresh then
5:              goto compute
6:          else
7:              if ptr.jobtype = refresh & matured(ptr) = true then
8:                  goto compute
9:              else
10:                 if ptr.jobtype = refresh & expired(ptr) = true then
11:                     dropped jobs ++
12:                     goto add
13:                 else
14:                     ptr ← next(ptr)
15:                 end if
16:             end if
17:         end if
18:     end while
19:     compute :
20:       compute shortest path
21:     add :
22:       find next hop
23:       add refresh job
24: end while
```

Job Queue Insertion

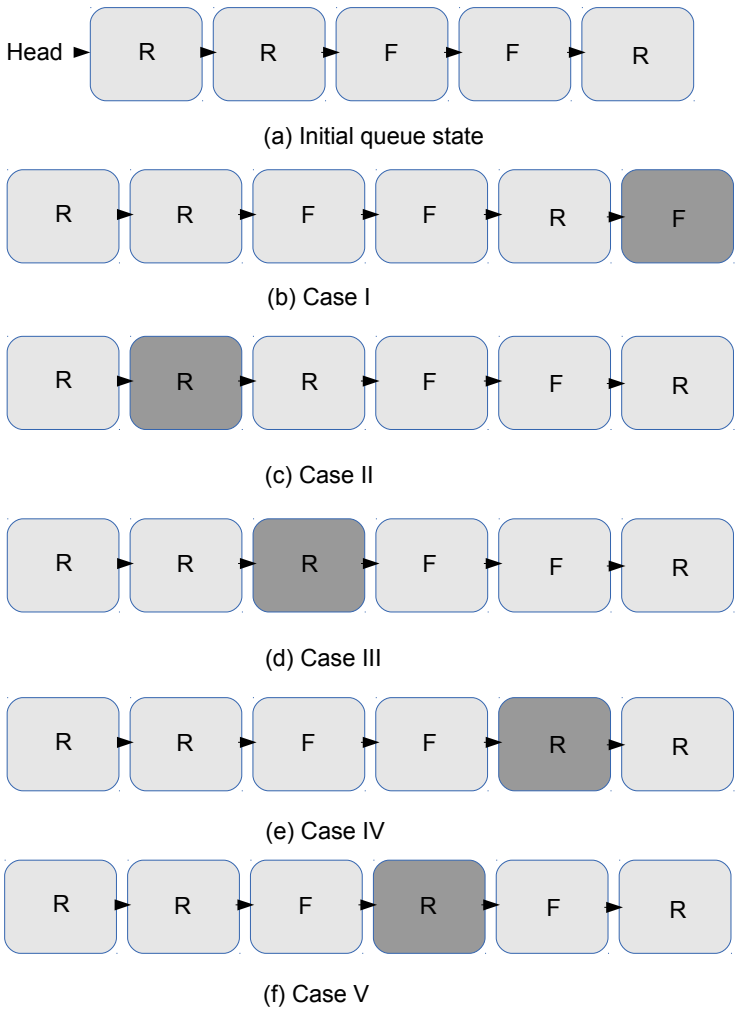


Job Queue Insertion



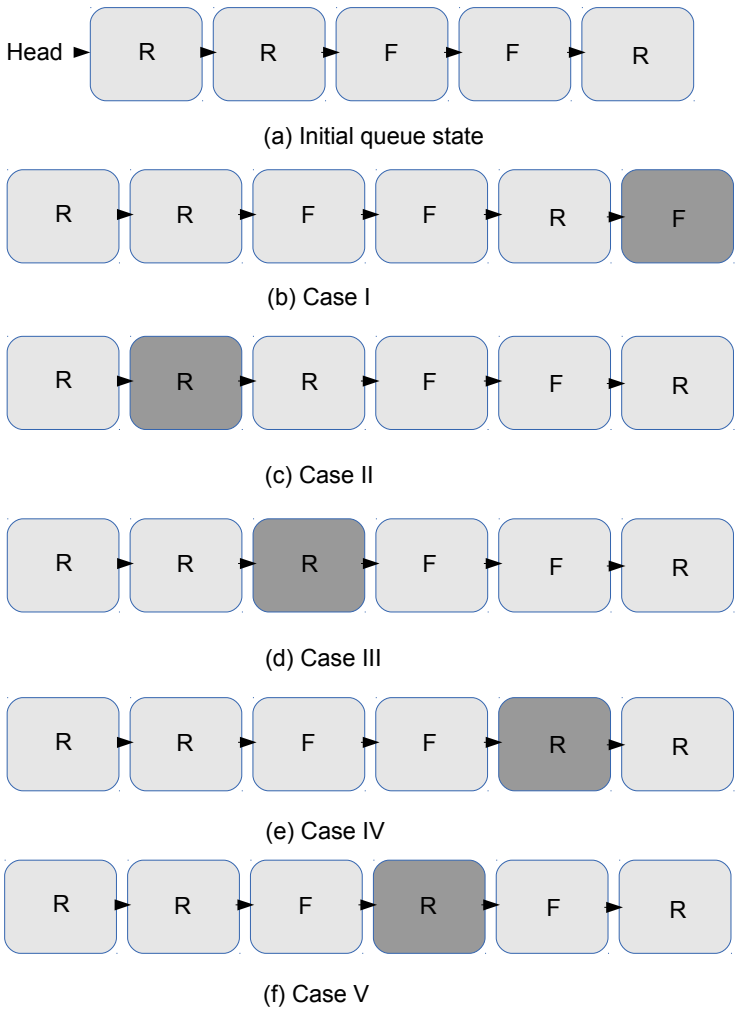
- Case I: Fresh job is simply added in the back

Job Queue Insertion



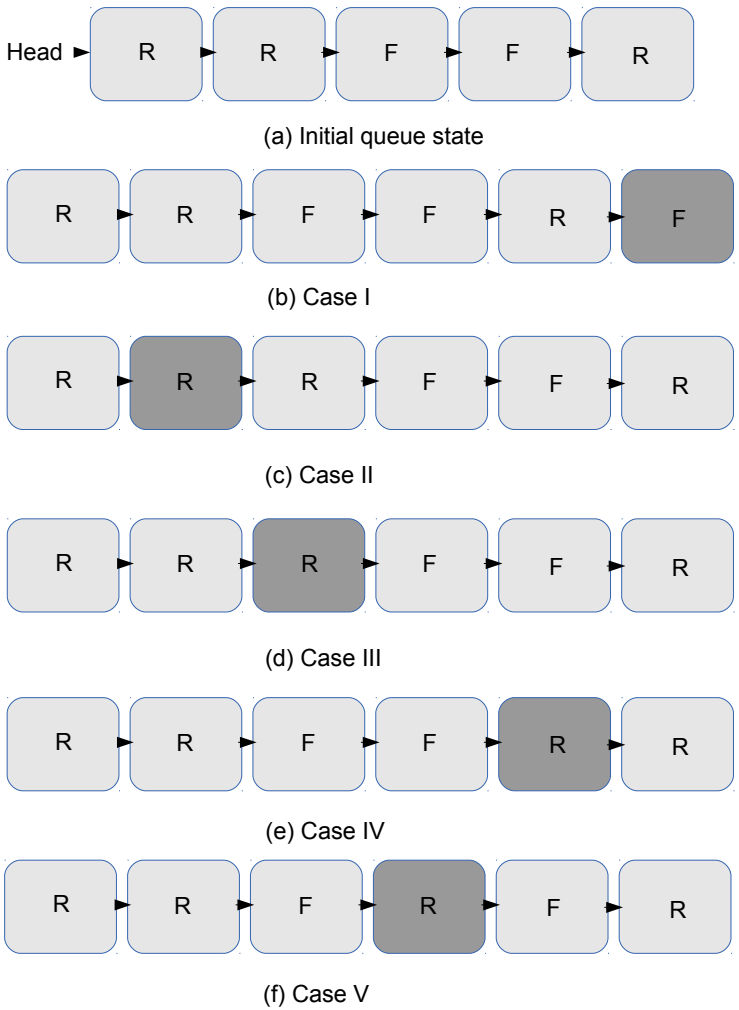
- Case I: Fresh job is simply added in the back
- Case II: If refresh job's Timestamp2 value falls between two consecutive refresh jobs, it is simply added in the middle

Job Queue Insertion



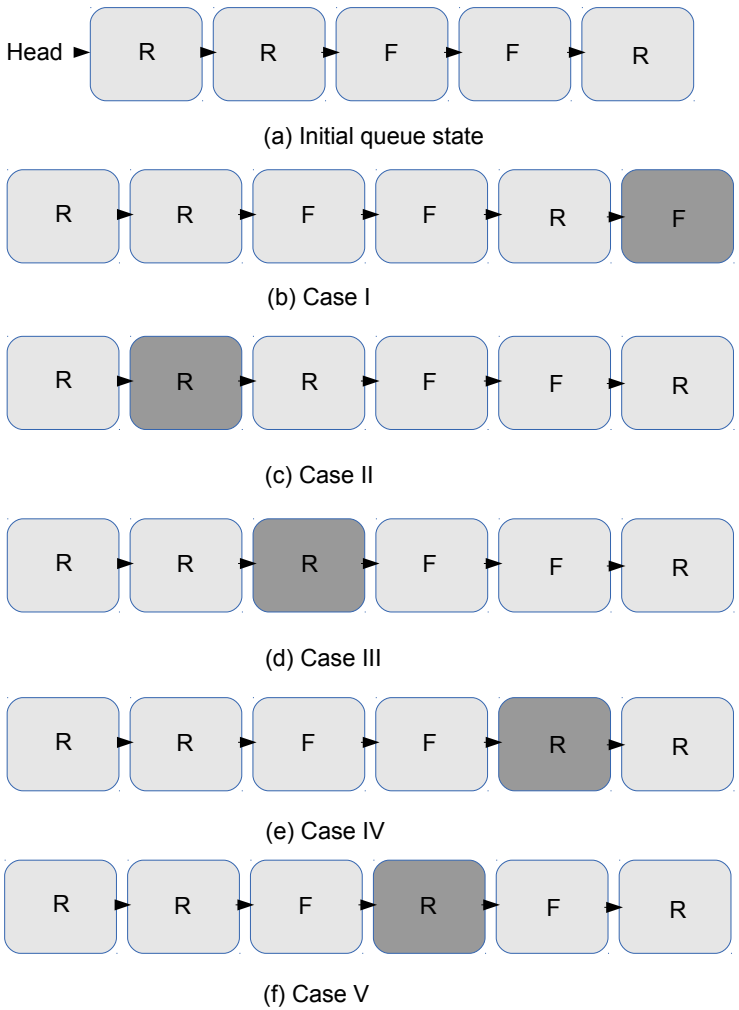
- Case I: Fresh job is simply added in the back
- Case II: If refresh job's Timestamp2 value falls between two consecutive refresh jobs, it is simply added in the middle
- Case III: Refresh job is given higher priority over fresh job.

Job Queue Insertion



- Case I: Fresh job is simply added in the back
- Case II: If refresh job's Timestamp2 value falls between two consecutive refresh jobs, it is simply added in the middle
- Case III: Refresh job is given higher priority over fresh job.
- Case IV: Fresh job is given higher priority.

Job Queue Insertion



- Case I: Fresh job is simply added in the back
- Case II: If refresh job's Timestamp2 value falls between two consecutive refresh jobs, it is simply added in the middle
- Case III: Refresh job is given higher priority over fresh job.
- Case IV: Fresh job is given higher priority.
- Case V: Relative priorities of fresh jobs and refresh jobs are dynamically computed.

Computing relative priorities for insertion

Computing relative priorities for insertion

If, $Timestamp2(R1) < Timestamp2(R2)$, then:

$$Priority(R1) > Priority(R2) \tag{4}$$

If, $AgingFactor(F) < Threshhold$, then:

$$Priority(R) > Priority(F) \tag{5}$$

If, $AgingFactor(F) \geq Threshhold$, then:

$$Priority(F) > Priority(R) \tag{6}$$

Algorithm for insertion of refresh job

```
1:  if (Queue Empty) then
2:      queue head ← next_refresh_job
3:      return done
4:  end if
5:  ptr ← queue head
6:  while ptr != NULL do
7:      if ptr.jobtype = refresh && ptr.timestamp2 > next_refresh_job.timestamp2
        then
8:          insert next_refresh_job before ptr
9:          return done
10:     end if
11:     if ptr.jobtype = fresh && ptr.age < threshold then
12:         insert next_refresh_job before ptr
13:         temp ← ptr
14:         while temp.jobtype = freshjob do
15:             temp.age++
16:             temp ← next(temp)
17:         end while
18:         return done
19:     end if
20:     ptr ← next(ptr)
21: end while
22: add next_refresh_job at queue end
23: return done
```

Optimize Route Computation per Query

- Selecting the best algorithm for a particular query.

Optimize Route Computation per Query

- Selecting the best algorithm for a particular query.
- We tried to do this by estimating the search space using the notion of graph density.

Optimize Route Computation per Query

- Selecting the best algorithm for a particular query.
- We tried to do this by estimating the search space using the notion of graph density.
- Most algorithms use Bidirectional search over simple Dijkstra's algorithm because of it's lesser avg. query computation time.

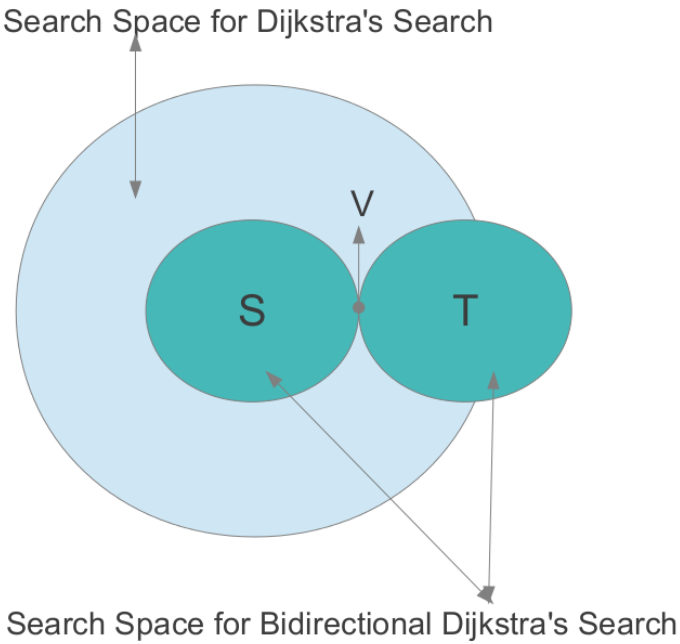
Optimize Route Computation per Query

- Selecting the best algorithm for a particular query.
- We tried to do this by estimating the search space using the notion of graph density.
- Most algorithms use Bidirectional search over simple Dijkstra's algorithm because of it's lesser avg. query computation time.
- Possible to select which of the two algorithms would be better for a given query.

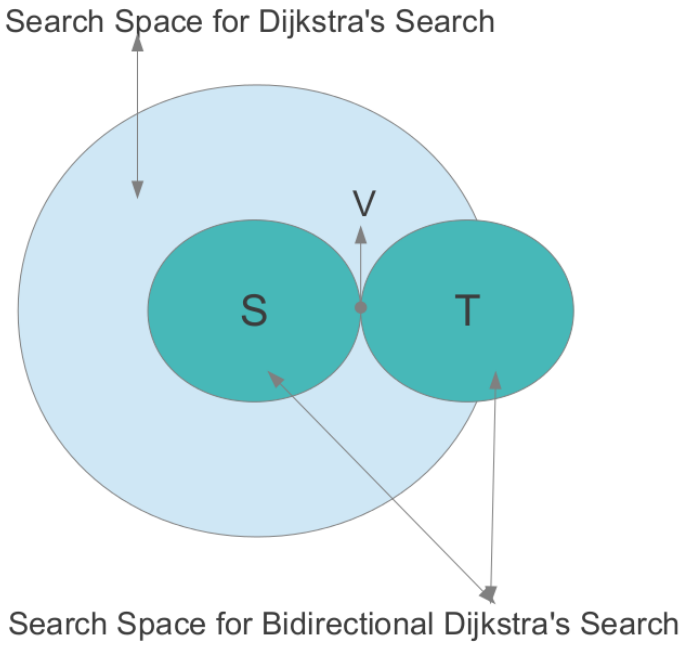
Optimize Route Computation per Query

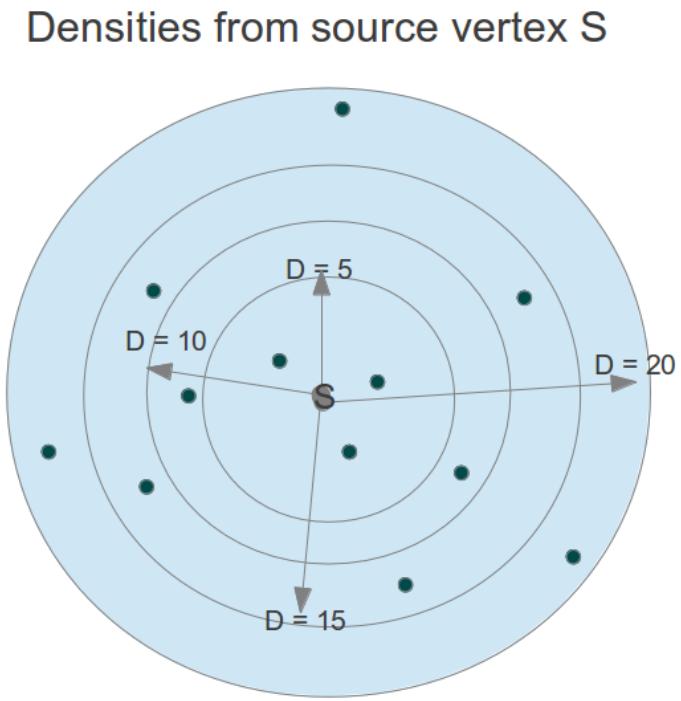
- Selecting the best algorithm for a particular query.
- We tried to do this by estimating the search space using the notion of graph density.
- Most algorithms use Bidirectional search over simple Dijkstra's algorithm because of it's lesser avg. query computation time.
- Possible to select which of the two algorithms would be better for a given query.
- Also reduces the average query computation time.

Search Space and Graph Density



Search Space and Graph Density





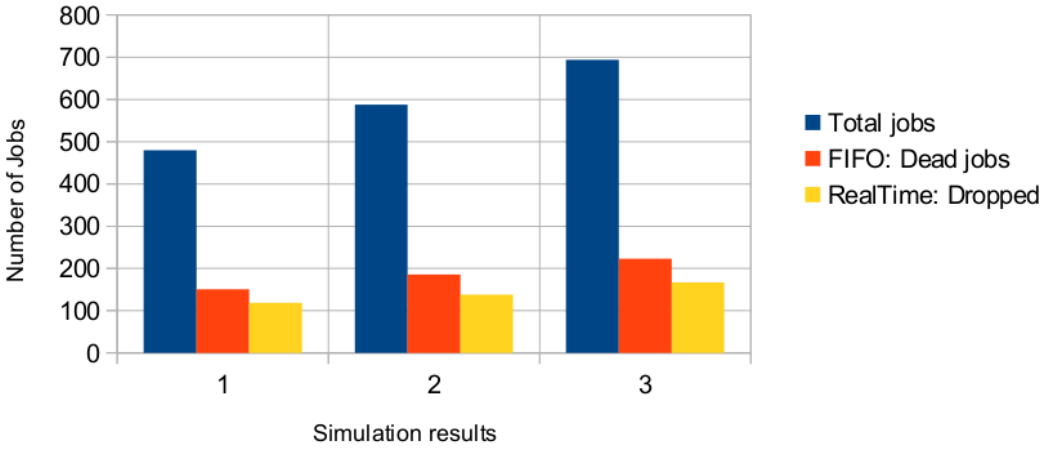
Experimental Setup

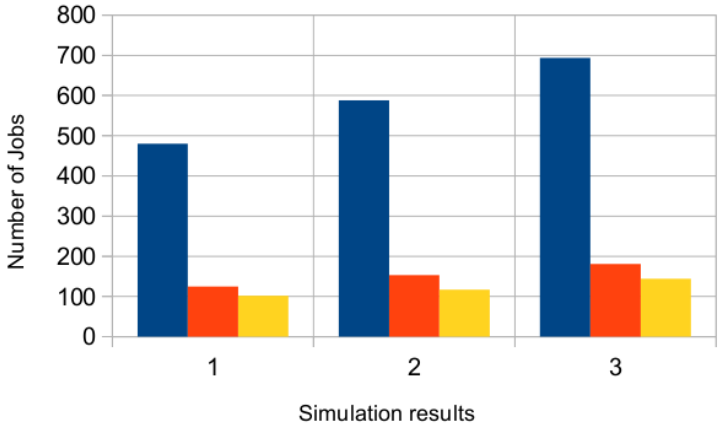
- Road networks were taken from Dimacs Implementation Challenge.

Table: Graph instances used

Graph Name	Number of Vertices	Number of Edges
TG	18263	23874
BAY	174956	223001
SF	321270	800172
COL	435666	1057066
LKS	2758119	6885658

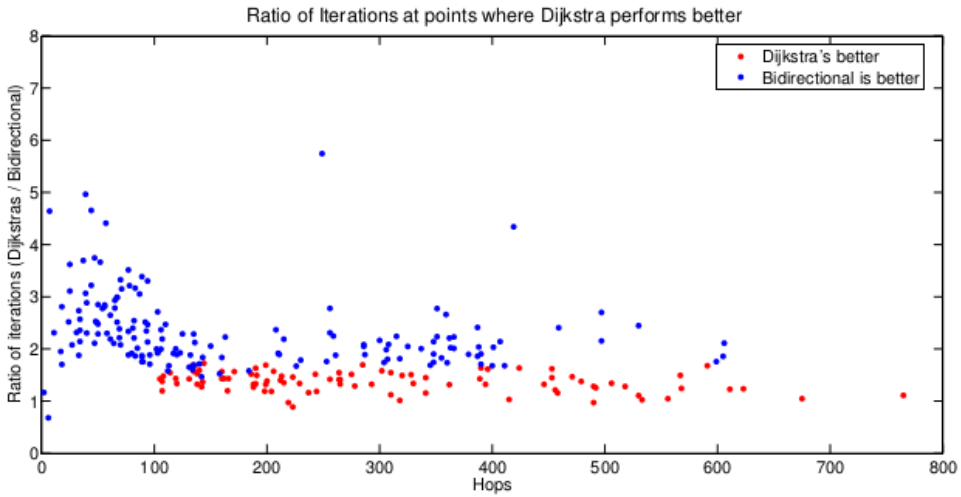
Results for Real Time Routing

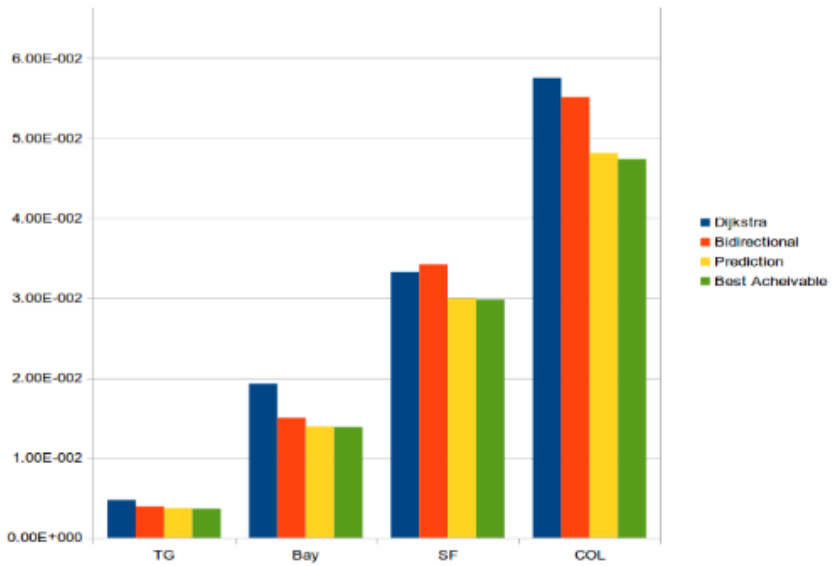




- Showing simulation results with $\alpha = 0.6$ (top) and 0.5 (bottom) and two threads.
- For $\alpha = 0.6$, 31-32% queries are dead in FIFO, 23-24% queries are dropped by our framework.
- With increasing α, scheduling window decreases and more number of queries result into dead/drop computation.
- Our framework processes 7-10% more number of useful queries and takes 10-12% lesser time than FIFO on an average.

Results for Query Dependent Route Computation





Timing results (in seconds) for various road networks

Future Work

- Experimentally coming up with optimal system parameters like α and aging factor.

Future Work

- Experimentally coming up with optimal system parameters like α and aging factor.
- Including driver dependent information like speed group categorization for varying α.

Future Work

- Experimentally coming up with optimal system parameters like α and aging factor.
- Including driver dependent information like speed group categorization for varying α.
- Turnaround time for fresh jobs vs. number of dropped jobs influence the aging factor.

Future Work

- Experimentally coming up with optimal system parameters like α and aging factor.
- Including driver dependent information like speed group categorization for varying α.
- Turnaround time for fresh jobs vs. number of dropped jobs influence the aging factor.
- Between Dijkstra's search and it's bidirectional variant, the nature of subgraph influences the computation time. We can categorize more algorithms using the underlying network structure in similar way.

Future Work

- Experimentally coming up with optimal system parameters like α and aging factor.
- Including driver dependent information like speed group categorization for varying α.
- Turnaround time for fresh jobs vs. number of dropped jobs influence the aging factor.
- Between Dijkstra's search and it's bidirectional variant, the nature of subgraph influences the computation time. We can categorize more algorithms using the underlying network structure in similar way.

This work has been submitted to the 4th IEEE International Conference on Big Data and Cloud Computing (BDCloud 2014).

References

1. E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik, vol. 1, no. 1, pp. 269â271, 1959
2. R. Geisberger, P. Sanders, D. Schultes, and D. Delling, Contraction hierarchies: Faster and simpler hierarchical routing in road networks. Springer, 2008, pp. 319â333.
3. H. Bast, S. Funke, P. Sanders, and D. Schultes, Fast routing in road networks with transit nodes, Science, vol. 316, no. 5824, pp. 566â566, 2007
4. J. Sankaranarayanan and H. Samet, Query processing using distance oracles for spatial networks, Knowledge and Data Engineering, IEEE Transactions on, vol. 22, no. 8, pp. 1158â1175, 2010.
5. Abraham, Ittai; Delling, Daniel; Goldberg, Andrew V, A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks, Symposium on Experimental Algorithms, pages 230-241, 2011
6. Wikipedia.org, Shortest path problem, Road networks
7. D. Delling, P. Sanders, D. Schultes, and D. Wagner, Engineering route planning algorithms, in Algorithmics of large and complex networks. Springer, 2009, pp. 117â139
8. Wikipedia.org, Automotive navigation system, Misdirection
9. Jing Yuan ; Yu Zheng ; Xing Xie, T-Drive: Enhancing Driving Directions with Taxi Drivers' Intelligence, IEEE Transactions on , vol.25, no.1, pp.220,232, Jan. 2013
10. Verroios, Vasilis and Kollias, Konstantinos and Chrysanthis, Panos K. and Delis, Alex, Adaptive Navigation of Vehicles in Congested Road Networks, ICPS '08
11. Boriboonsomsin, K.; Barth, M.J.; Weihua Zhu; Vu, A., âEco-Routing Navigation System Based on Multisource Historical and Real-Time Traffic Information,â Intelligent Transportation Systems, IEEE Transactions on , vol.13, no.4, pp.1694,1704, Dec. 2012

Thank you.