Spatial Crowdsourcing: Task Assignment & Scheduling

Cyrus Shahabi, Ph.D.
Professor of Computer Science & Electrical Engineering
Director, Integrated Media Systems Center (IMSC)
Viterbi School of Engineering
University of Southern California
Los Angeles, CA 900890781
shahabi@usc.edu

OUTLINE

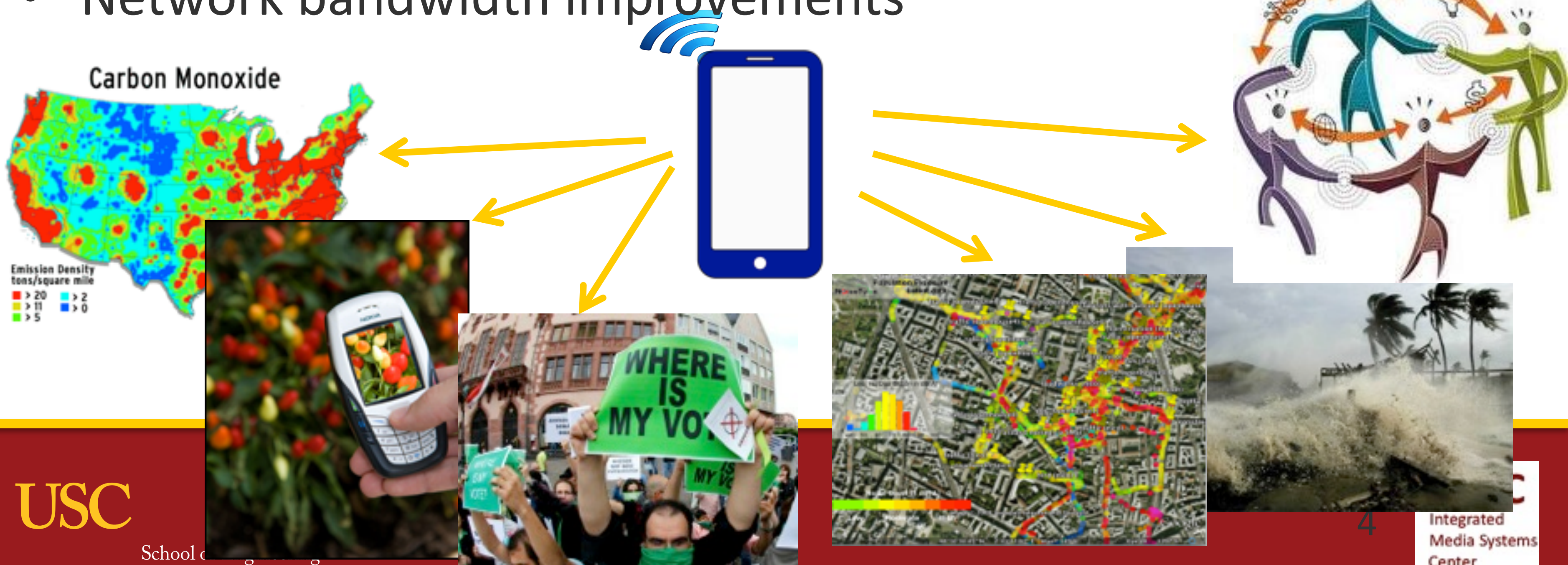
- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

OUTLINE

- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

Motivation

- Ubiquity of mobile users
  - 6 billion mobile subscriptions by the end of 2011
    ≡ 87% of the world population[1]
- Technology advances on mobile phones (e.g., Cameras)
- Network bandwidth improvements



[1] http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats

Spatial Crowdsourcing [ACMGIS'12]

Crowdsourcing: outsourcing a set of tasks to a set of workers. amazon mechanical turk™ Artificial Artificial Intelligence

Spatial crowdsourcing (SC)*:* requires workers to ***physically*** travel at the task's location in order to execute the task.

SC Applications

Ubiquity of mobile users ← **6.5 billion** mobile subscriptions, 93.5% of the world population [1]

Technology advances on mobiles ← Smartphone's **sensors**. e.g., video cameras

Network bandwidth improvements ← From 2.5G (up to 384Kbps) to 3G (up to 14.7Mbps) and recently **4G** (up to 100 Mbps)

[1] http://w... newsroom/id/2665715

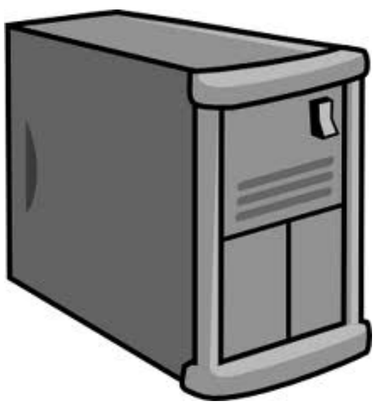
IMSC

GeoCrowd

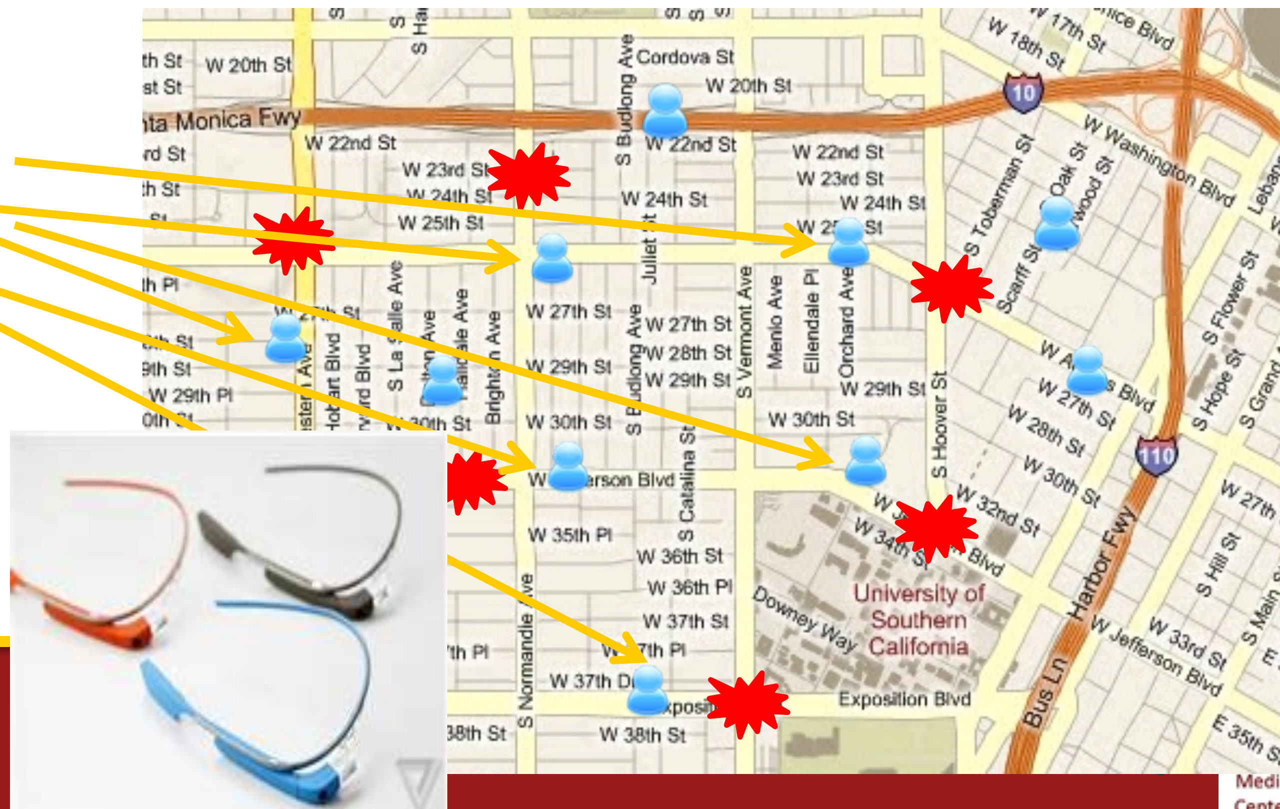
Worker

Requester

In Collaboration w Prof. Zimmermann, NUS



Spatial Crowdsourcing Server (SC-server)



NORTHROP GRUMMAN

Google

NSF

USC

School of Engineering

Integrated Media Systems Center

OUTLINE

- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

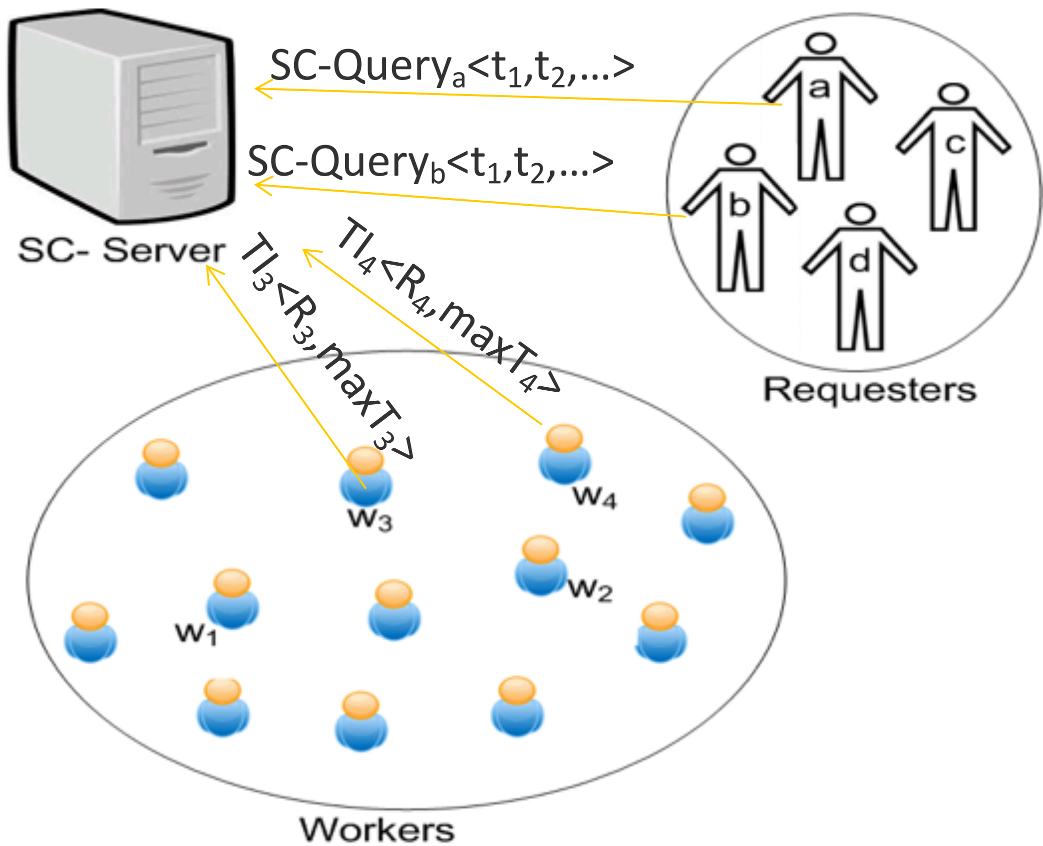
Problem Definition

- **Input:** Given *m spatial task sets* and *k workers*
- **Output:** assign *spatial task sets* to *workers* and provide *schedules* of workers
- **Objective:** that minimize the *total cost* (or maximize the *number of assigned tasks*) under *time / order constraints*.
- **Challenges:** An OR problem with: scale (DB), online (Algo), dynamism (Control), spatial (Geo), etc.

Preliminaries

- *Spatial task t<d, l, s, δ>:* Task *t* with description *d* to be answered at location *l*, asked at time *s* and will be expired at time *s+δ*.

- *Spatial Crowdsourced Query (SC-Query) $<t_1, t_2, ..>$:* A set of spatial tasks issued by a requester to the SC-server for crowdsourcing.
  - *Task Inquiry TI<R,maxT>:* Request that a worker *w* sends to the SC-server when ready to work with constraints:
    - *R*: A spatial region (e.g., rectangle) in which *w* accepts tasks
    - *maxT*: Maximum number of tasks *w* can perform



Problem Definition

- *Task Assignment Instance Set I_i*
  - $W_i = \{w_1, w_2, \ldots\}$: Set of workers at time s_i
  - $T_i = \{t_1, t_2, \ldots\}$: Set of available tasks at time s_i
  - $I_i = \{<w,t> \mid w \in W_i,\ t \in T_i\}$: a spatial task *t* is assigned to a worker *w*, while satisfying the workers' constraints.
- *Maximum Task Assignment (MTA)*
  - $\phi = \{s_1, s_2, \ldots, s_n\}$: A time interval
  - *MTA:* Maximizing the total number of assigned tasks during ϕ while satisfying the workers' constraints
    - Maximizing $\sum_{i=1}^{n} |I_i|$

USC School of Engineering

IMSC Integrated Media Systems Center

Related Work

- Crowdsourcing
  - Services/Markets/App
    - Amazon's Mechanical Turk (MTurk)
    - CrowdFlower, oDesk, Waze
  - Research
    - Databases [MIT, Stanford, Berkeley]
    - Data Analytics [Liu et al. and Wang et al., PVLDB'12]
    - Image search [Yan et al., MobiSys'10]
    - Natural language annotations [Snow et al., EMNLP'08]
    - Social games [Guy et al., CHI'11]
    - Search [Alonso et al., SIGIR'11]

Non-Spatial

- Spatial Crowdsourcing
  - [Alt et al., NordiCHI'10] → Worker Selected Spatial Crowdsourcing (application)
  - [Bulut et al., PerCom Workshops'11] → Non-spatial tasks
- Participatory Sensing: An instance of spatial crowdsourcing in which there is only one requester (i.e., campaign) and tasks are only sensing tasks
  - CENS
  - [Hull et al., SenSys'06]
  - [Mohan et al., SenSys'08]
  - [Cornelius et al., MobiSys'08]
  - [Shirani-mehr et al., GIS'09]

✓Not focused on task assignment

✓ Focus on single campaign

✓ Not a general framework

- Volunteered Geographic Information (VGI) : Create geographic information provided voluntarily by individuals
  - StreetMap
  - Google Map Maker
  - WikiMapia

✓ Users unsolicited participation by randomly contributing data

Related Work (Task Assignment)

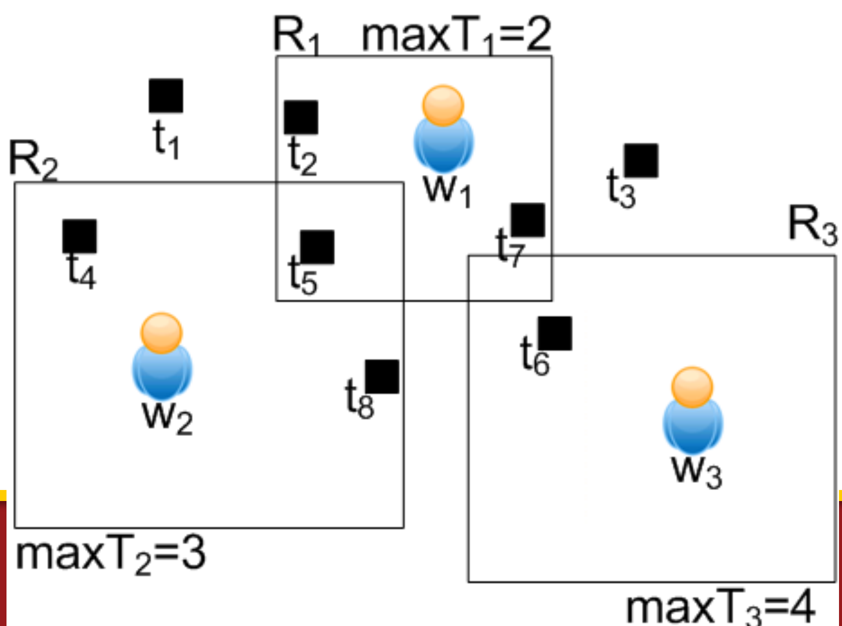
- Classic Matching problems – matching tasks w workers
- Real-time matching – [Kalyanasundaram and Pruhs, 1993 & 2000]
  - Spatial characteristic of tasks and workers
    - Adding spatial feature as a metrics increase complexity (not scalable)
- Spatial matching – [Wong, Tao, Fu and Xiao, 2007][U, Yiu, Mouratidis and Mamoulis, 2008]
  - Dynamism of tasks and workers (i.e., tasks and workers come and go without our knowledge),
    - The challenge is to perform the task assignment at a given instance of time with the goal of global optimization across all times
  - Workers need to travel to task locations causes the landscape of the problem to change constantly
    - This will add another layer of dynamism to spatial crowdsourcing that makes it a unique problem

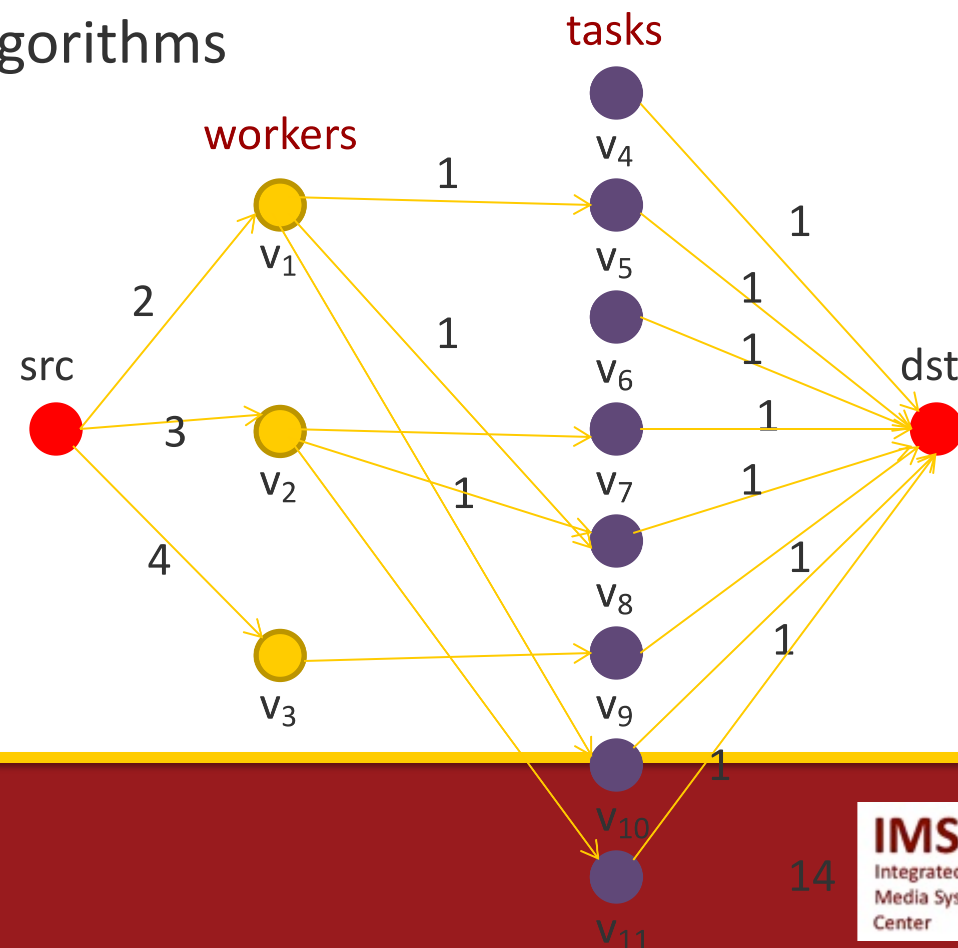
Assignment Protocol

- Future knowledge → Optimal assignment → Solving MTA
- Challenge
  - Current knowledge at every time instance → Local optimization
- Goal
  - Optimizing the task assignment locally by utilizing the spatial information that workers share during their task inquiries
- Approaches
  - Greedy (GR) Strategy
  - Least Location Entropy Priority (LLEP) Strategy
  - Nearest Neighbor Priority (NNP) Strategy

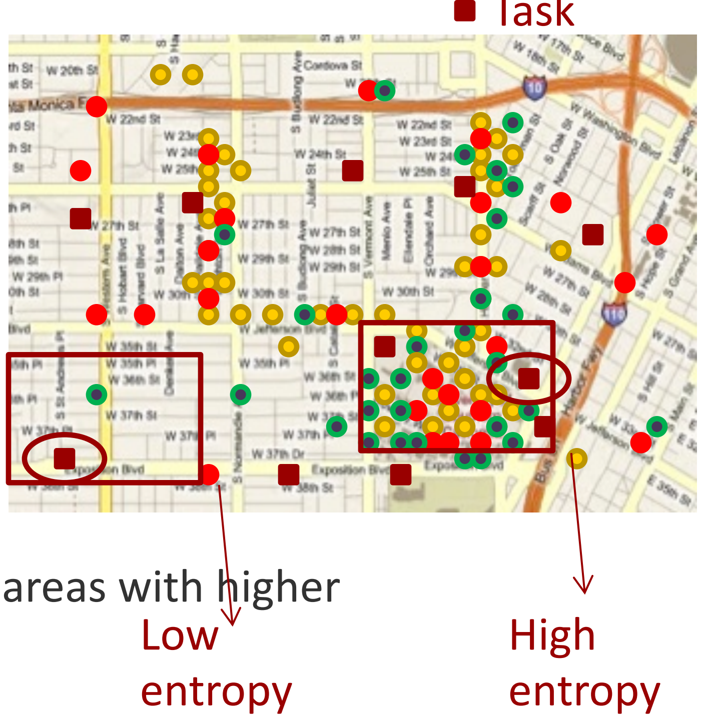
Greedy (GR) Strategy

- Goal → Maximizing the assignment at every instance of time s_i → solving Maximum Task Assignment Instance ($MTAI_i$)
- $MTAI_i$ is equivalent to max-flow problem
- Apply any of the max-flow algorithms
  - Ford-Fulkerson





Least Location Entropy Priority Strategy (LLEP)



- Goal
  - Exploiting the spatial characteristics of the environment to maximize the overall task assignment
- Intuition
  - A task is more likely to be completed when located in areas with higher worker densities
- Heuristic
  - Assigning higher priority to tasks which are located in worker-sparse areas
- Location Entropy: Measuring the total number of workers in a location as well as the relative proportion of their visits to that location
  - l : location
  - O_l : Set of visits to location l
  - W_l : Set of distinct workers that visited l
  - $O_{w,l}$: Set of visits that worker w has made to the location l
  - $P_l(w) = |O_{w,l}| / |O_l|$

$$Entropy(l) = -\sum_{w \in W_l} P_l(w) \times \log P_l(w)$$

Major Observations

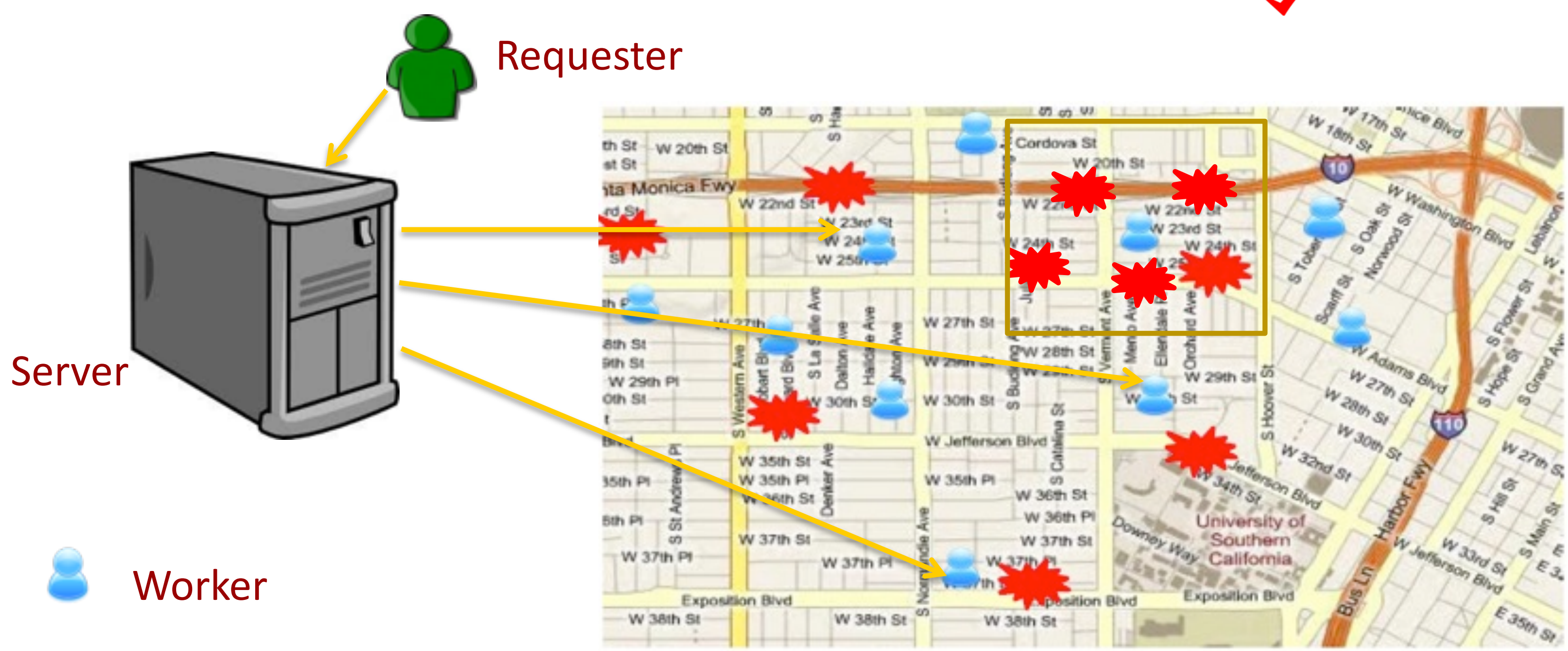
- Experiments on both real and synthetic data demonstrated
  - The superiority of LLEP in comparison with GR in terms of the number of assigned tasks by up to 36%

OUTLINE

- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

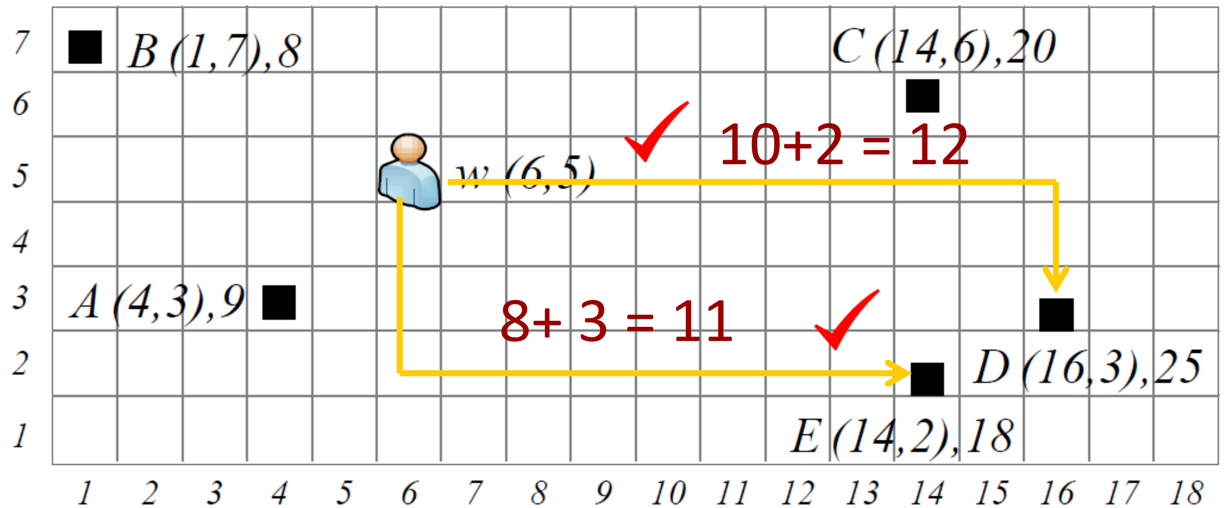
Spatial Crowdsourcing

- Challenges:
  - Task Assignment
- Approach
  - Server Assigned Tasks (SAT) V.S Worker Selected Tasks (WST)



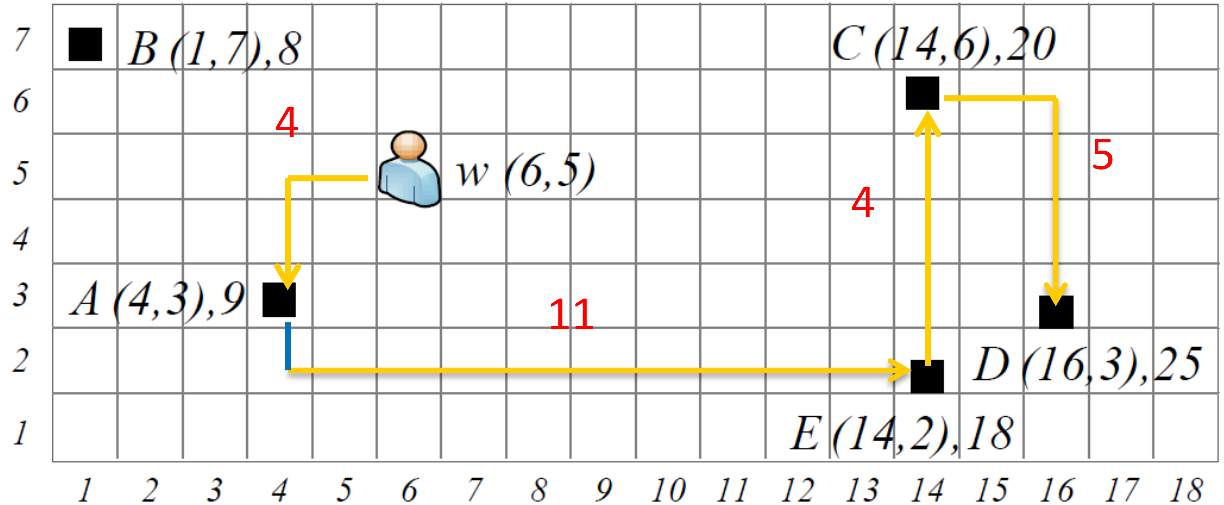
Example

- Tasks with location and deadlines
  - E.g. task D needs to be finished in 25 minutes
- Suppose travel time for one grid is one minute
  - Consider Manhattan distance
  - E.g., cost(w, D) = 10 + 2 = 12 minutes



Problem Definition

- Maximum Task Scheduling (MTS)
  - Given a worker *w* and a set of *n* tasks *T with locations and deadlines*
  - Find a maximal task sequence *R*



Maximal Task sequence: (A, E, C, D)

Problem Complexity

- The MTS problem is NP-hard by reduction from Traveling Salesman Problem (TSP)

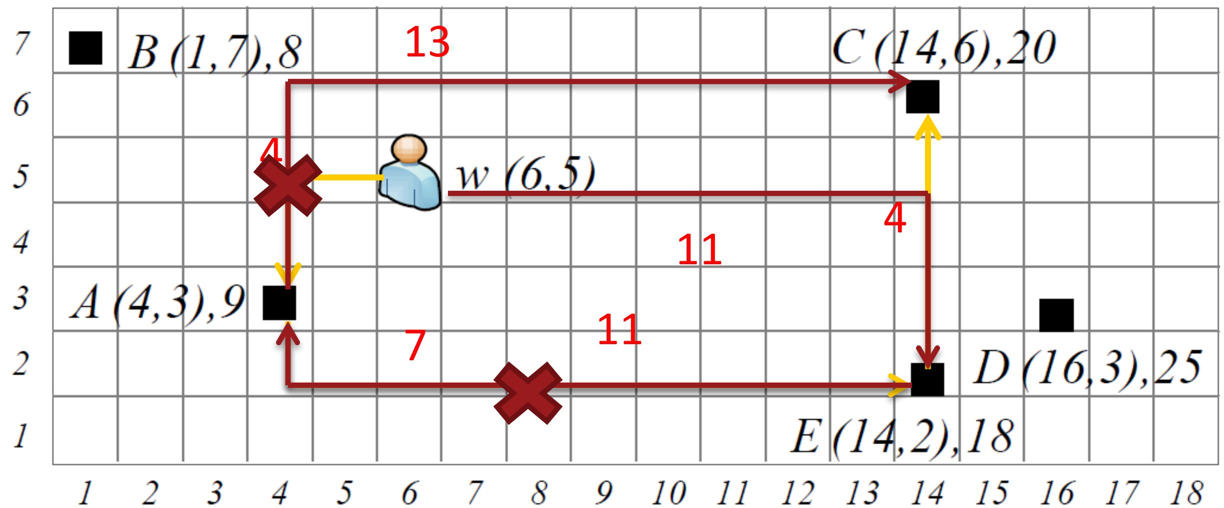
- Brute force takes O(n!) time

Outline

- Problem Definition
- **Exact Algorithms**
  - Dynamic Programming
  - Branch-and-Bound Algorithm
- Approximation Algorithms

Dynamic Programming

- Let's schedule task set *{A, C, E}* s.t. it ends w *C*
  - Schedule *{A, E}* ends with *E* → *3*
  - Or schedule *{A, E}* ends with *A* → *1*
  - *Choose the best among them*



Dynamic Programming

- Define *opt(S, j)* as the optimum number by scheduling all the jobs in S, ends with *j*
  - *Task i* is the *second-to-last* finished task before *j*

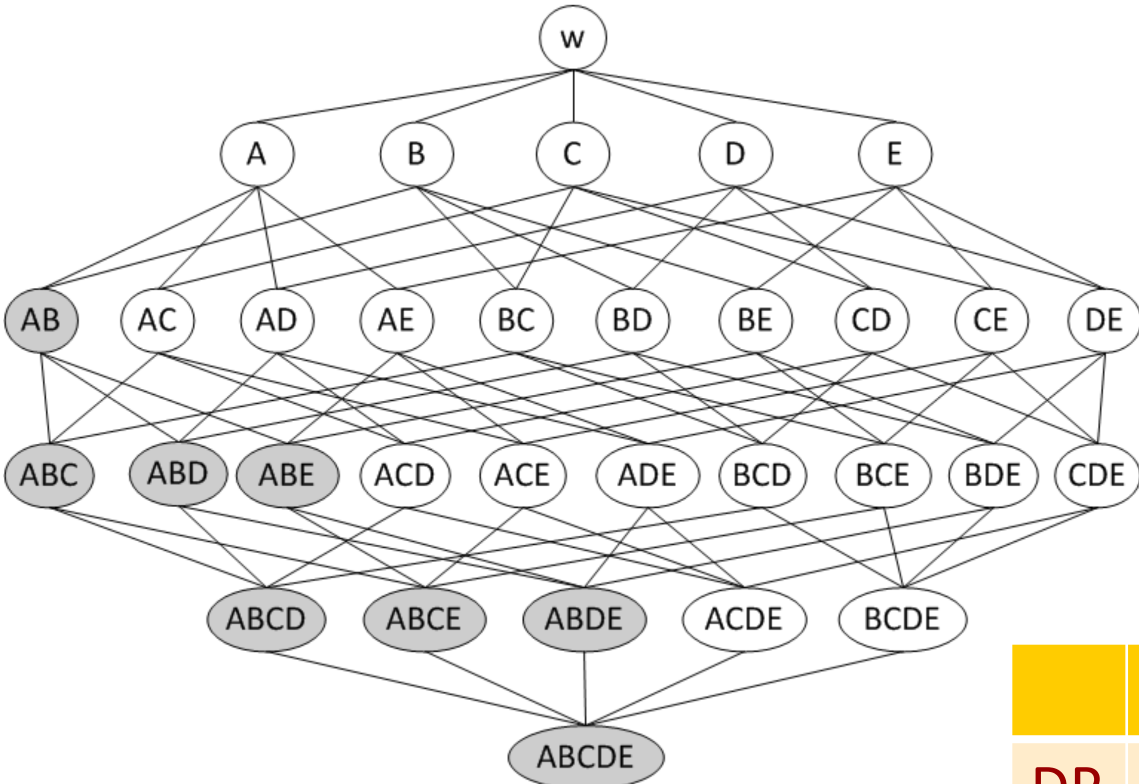
$$opt(S,j) = \max_{i \in S, i \neq j} \left(opt(S - \{j\}, i) + \delta_{ij} \right)$$

$$\delta_{ij} = \begin{cases} 1 & \text{if job } j \text{ can be finished after job } i \\ 0 & \text{else} \end{cases}$$

24

Dynamic Programming

- Subsets needs to be explored



$$\binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n} = 2^n$$

subsets in total

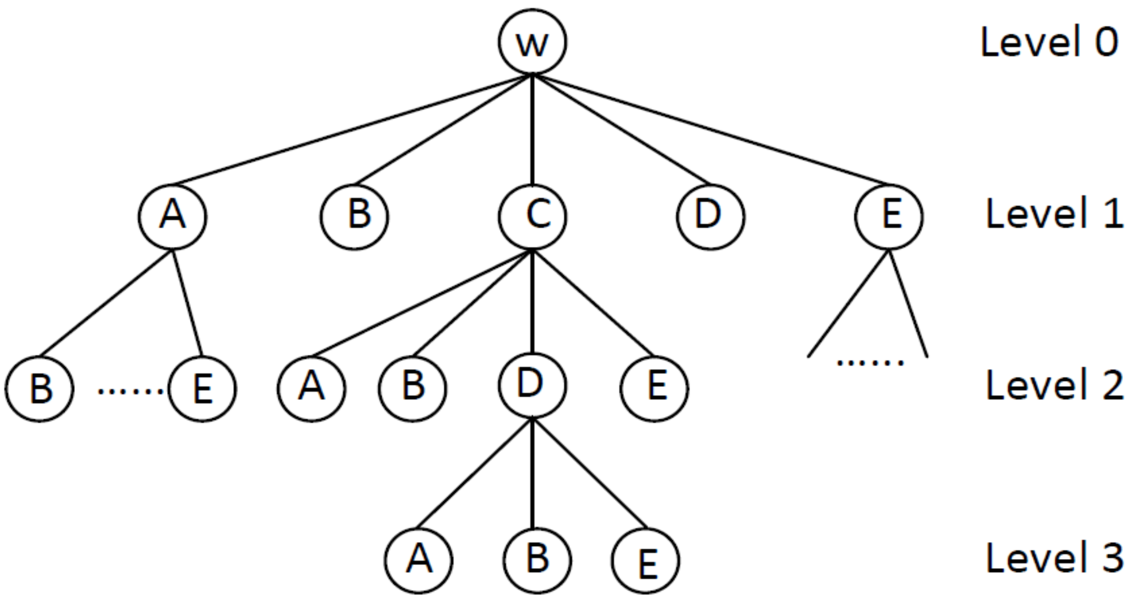
	Space Cost	Time Cost
DP	$O(n \bullet 2^n)$	$O(n^2 \bullet 2^n)$

Outline

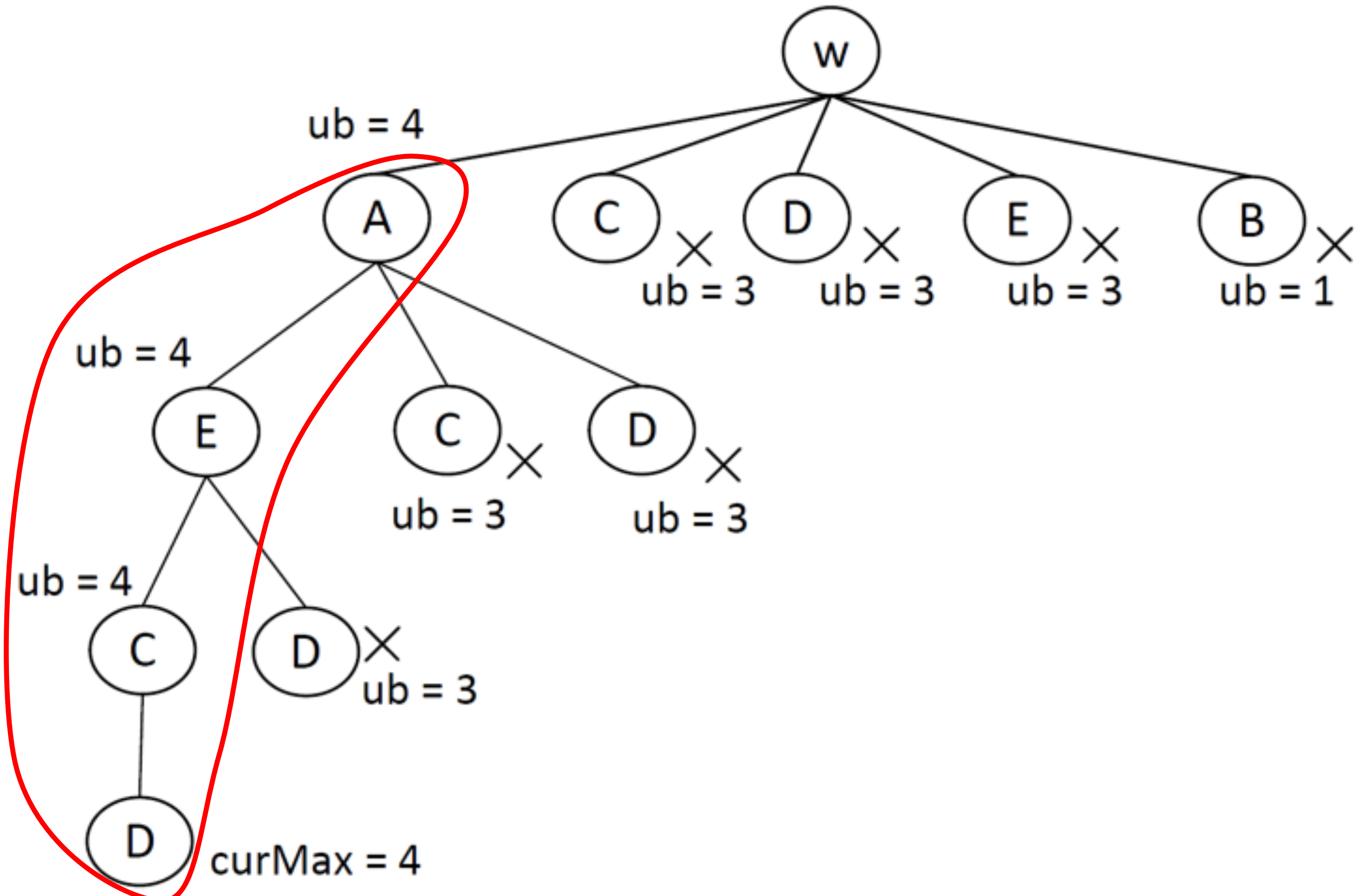
- Problem Definition
- **Exact Algorithms**
  - Dynamic Programming
  - **Branch-and-Bound Algorithm**
- Approximation Algorithms

Branch-and-Bound

- Search Tree
  - Depth-first or best-first search



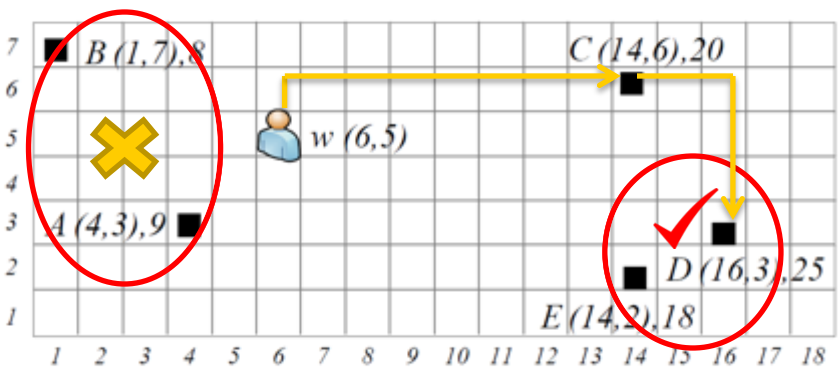
Example of B&B

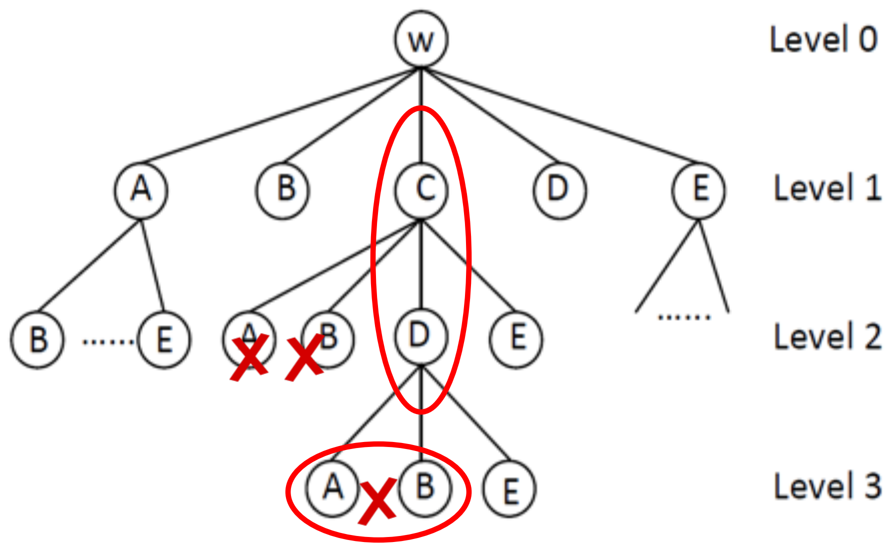


31

Candidate Task Set

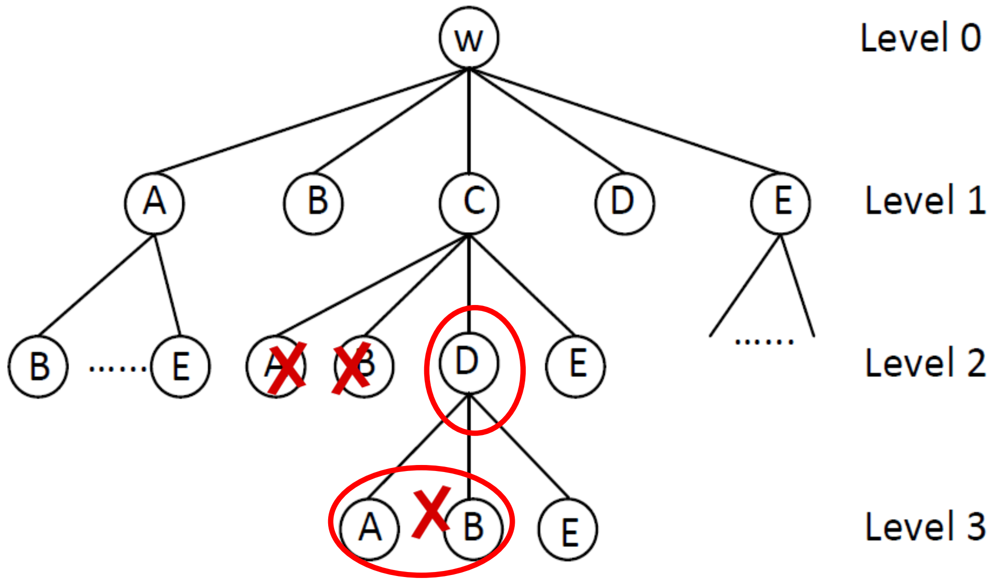
- Suppose we are at (C, D), do we still need to try A, B at level *3*?





Candidate Task Set (*cand*)

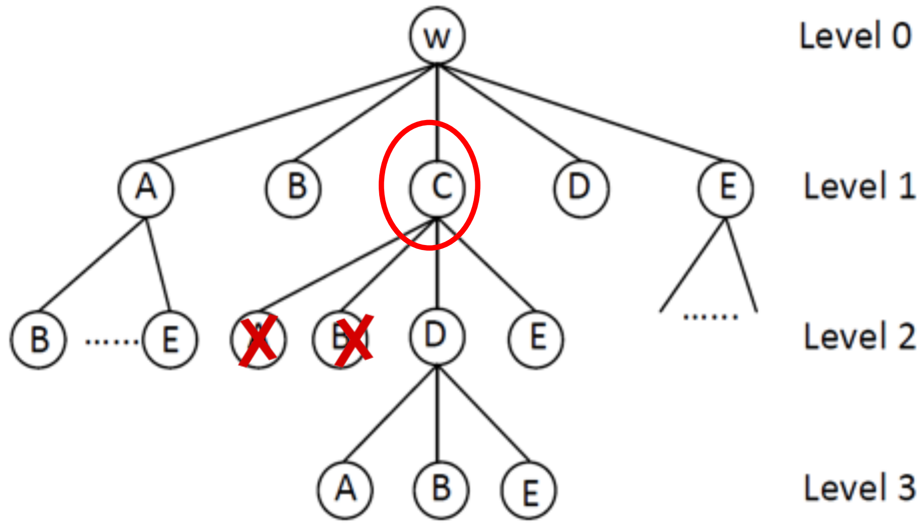
- A candidate task set maintains the promising tasks to be expanded at the next level:
  - e.g. *cand*(C) = {D,E}   *cand*(C, D) = {E}



- **Property**: A node's candidate tasks set is the subset of its parent's candidate task set

Bound of Branch

- R is current path from w
- Upper-bound of R
  - – $ub(R) = |R| + |cand\_R|$
  - – E.g., $ub(C) = 1 + 2 = 3$



- Lower-bound of R
  - – Minimum number of tasks that can be completed by following this branch

Branch-and-Bound

- Complexity

	Space Cost	Time Cost
DP	$O(n \bullet 2^n)$	$O(n^2 \bullet 2^n)$
B&B	$O(n^2)$	$O(n!)$

Worst case

- In reality, n is the number of tasks in the vicinity of the worker, it might be very large!

Outline

- Problem Definition
- Exact Algorithms
  - Dynamic Programming
  - Branch-and-Bound Algorithm
- **Approximation Algorithms**

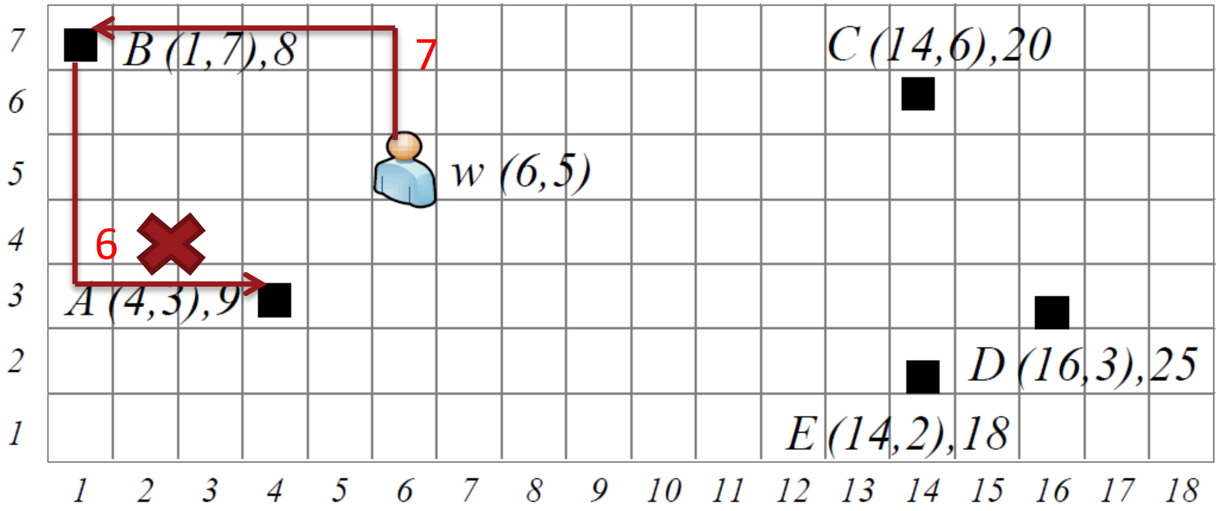
Limitation of Exact algorithms

- Restriction of Mobile platform
  - Limited CPU and memory resources
  - Interactive environment for the user
    - Response in milliseconds level

- Exact algorithms cannot scale
  - Exponential running time and/or huge memory consumption

Least Expiration Heuristics (LEH)

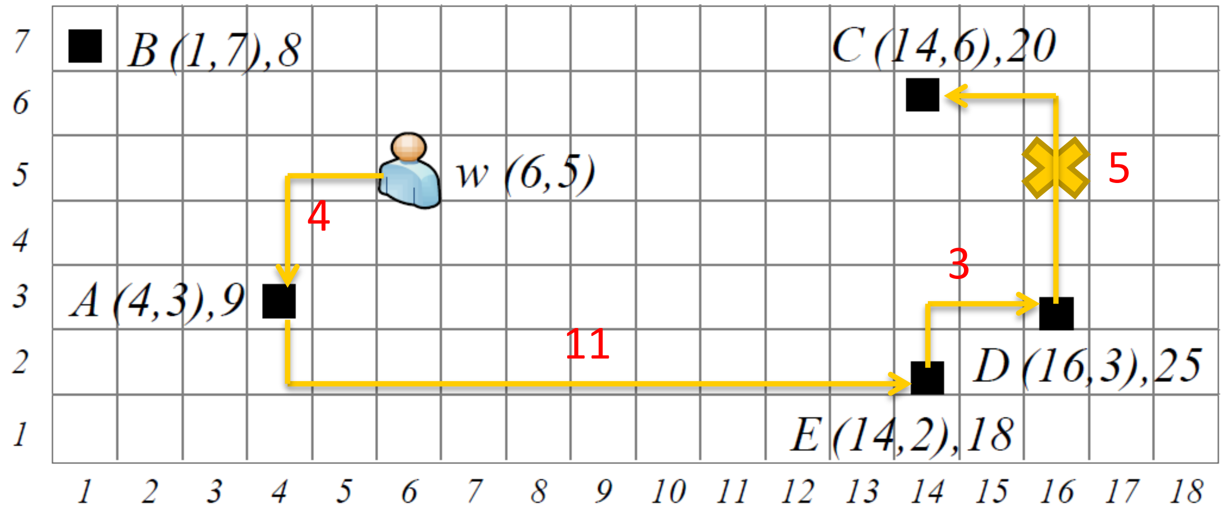
- Greedily choose the task with least expiration time



Task Sequence: <B>

Nearest Neighbor Heuristics (NNH)

- Greedily choose the task nearest to the worker

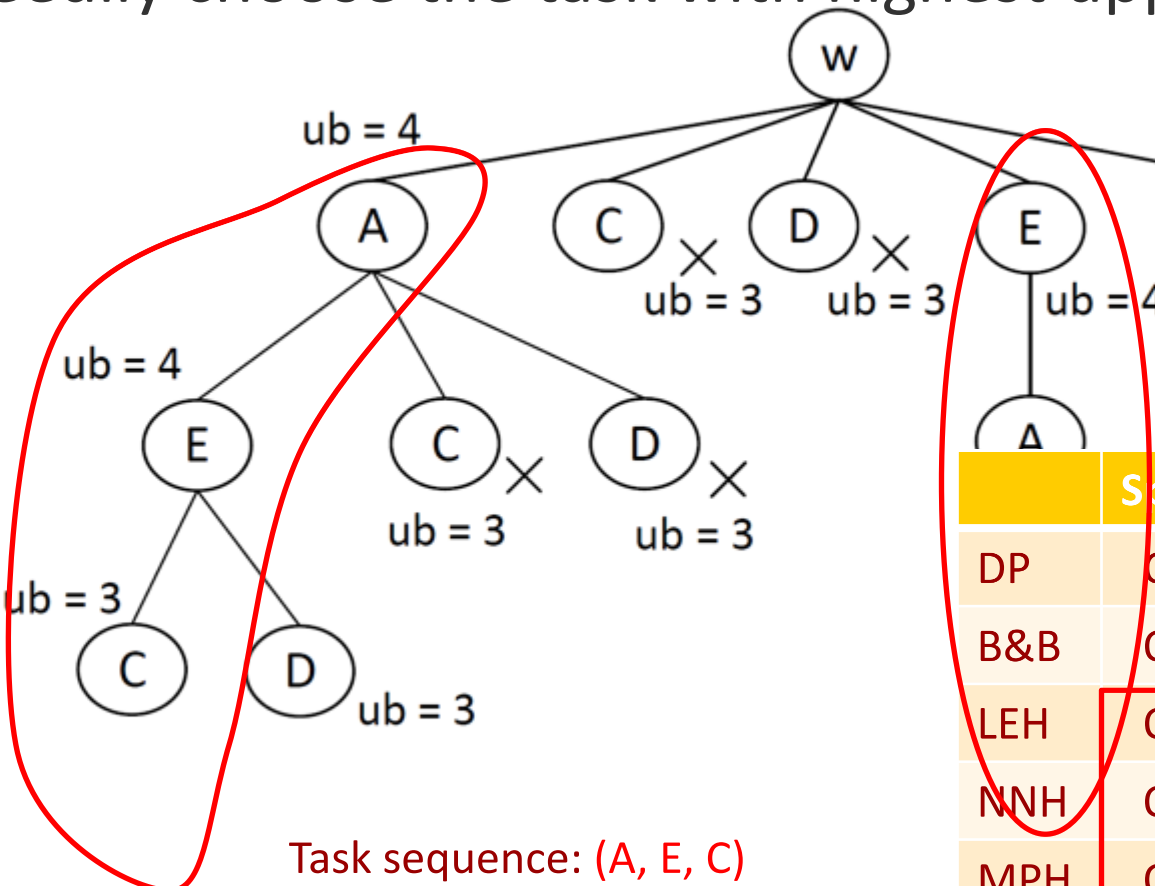


Task sequence: (A, E, D)

39

Most Promising Heuristic (MPH)

- Greedily choose the task with highest upper-bound

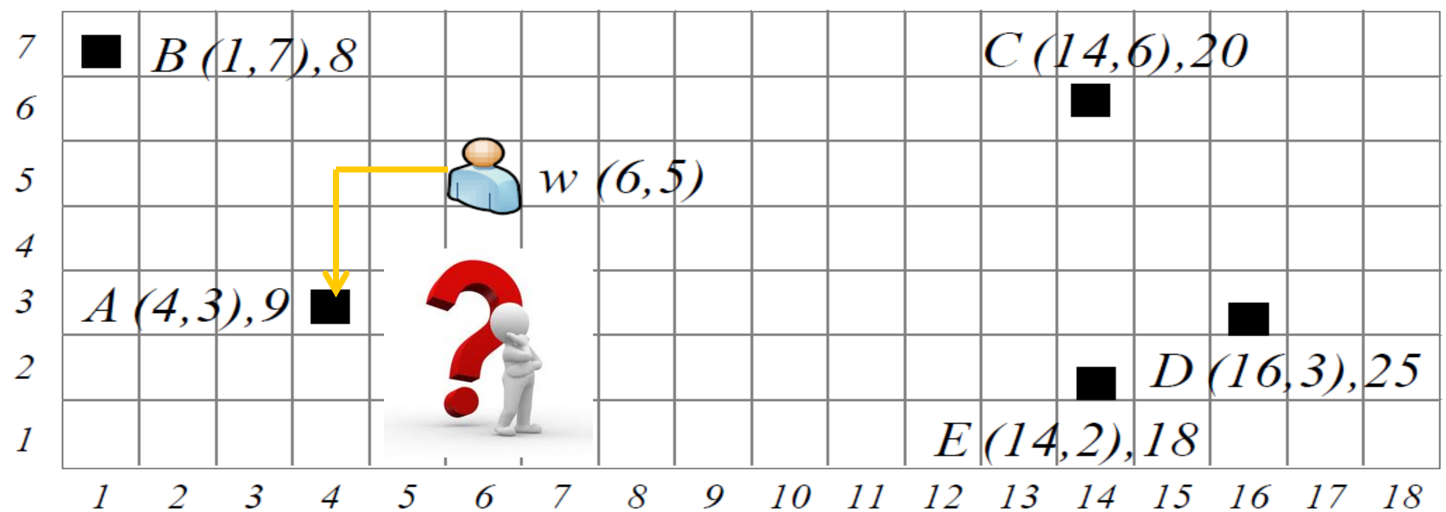


Task sequence: (A, E, C)

	Space Cost	Time Cost
DP	$O(n \cdot 2^n)$	$O(n^2 \cdot 2^n)$
B&B	$O(n^2)$	$O(n!)$
LEH	$O(1)$	$O(n \cdot \log(n))$
NNH	$O(1)$	$O(n^2)$
MPH	$O(n)$	$O(n^2)$

Progressive algorithms

- Approximation algorithm + Exact algorithm
  - NNH to choose the first task
  - Branch and Bound for the remaining *4* tasks



Progressive Algorithms

- Pros
  - Quick response time
  - Near-optimum results

- Cons
  - Preemption of other workers
  - Worker may prefer the whole plan

42

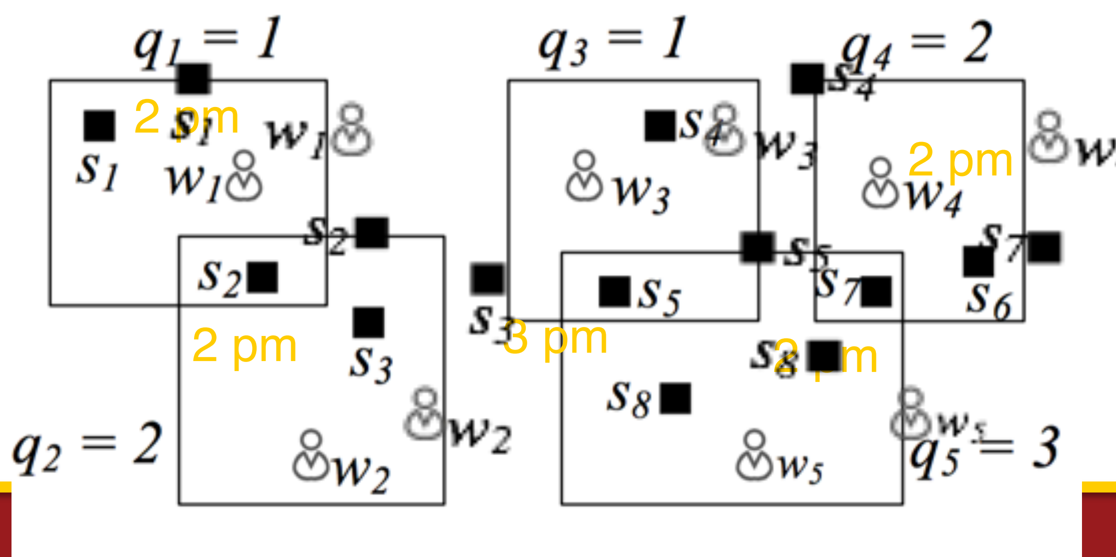
OUTLINE

- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

Problem definition

Input: Given a set of workers ***W*** and a set of tasks ***S***

- worker: **spatial and capacity constrain**
- task: **expiration time constraint**



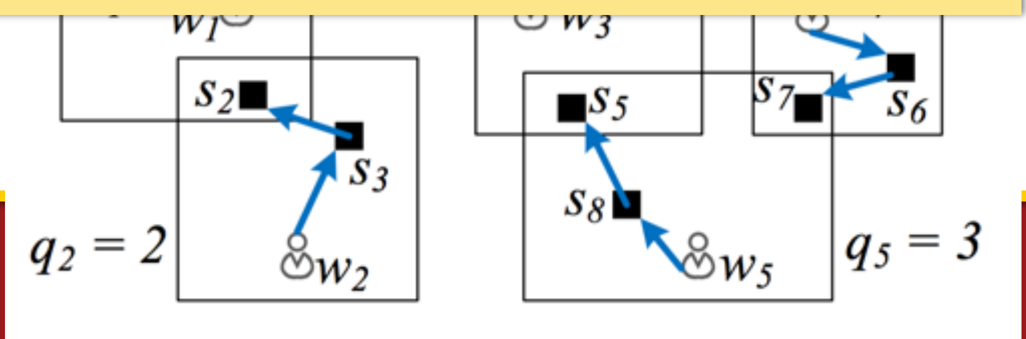
Problem definition

Input: Given a set of workers ***W*** and a set of tasks ***S***

Goal: find a scheduling plan for each worker:

1. Maximize the number of completed tasks (**primary goal**)
2. Minimize the average travel cost per task (**secondary goal**)

NP-hard Problem



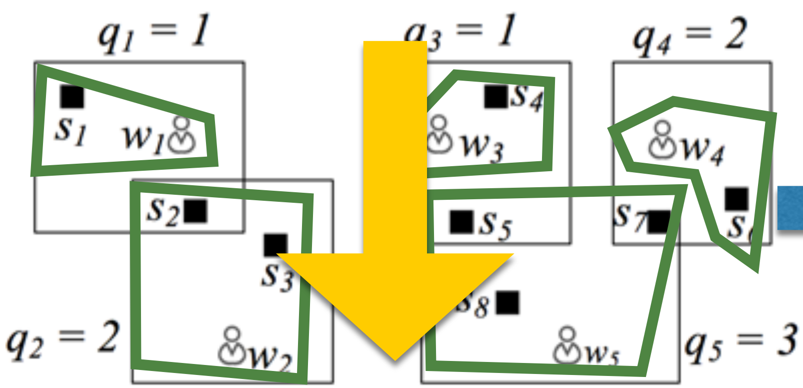
One potential plan

Outline

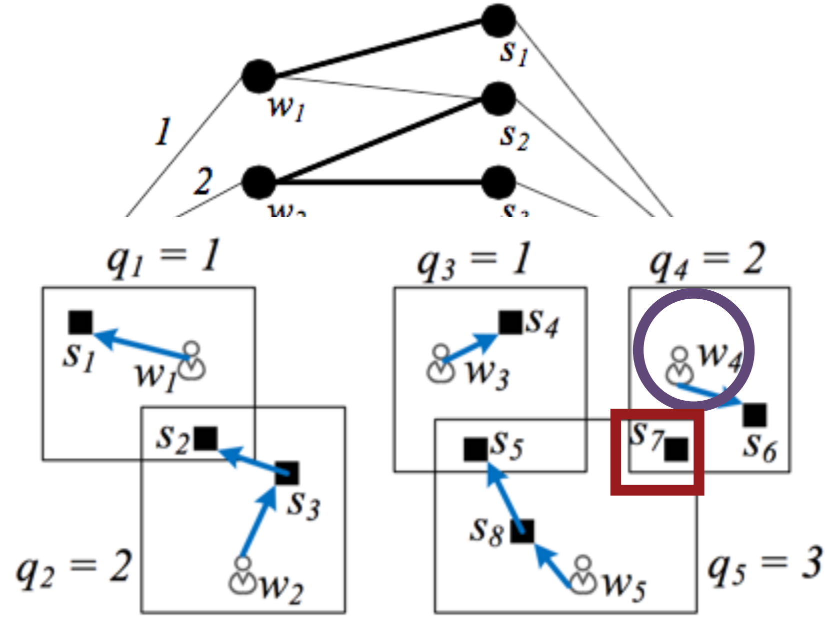
- **Global Assignment and Local Scheduling (GALS)**
- Local Assignment and Local Scheduling (LALS)
- Experiments

Baseline

1. Assignment via max-flow [Kazami'GIS12]



2. Schedule for each worker [Deng'GIS13]



Initial Assignment:

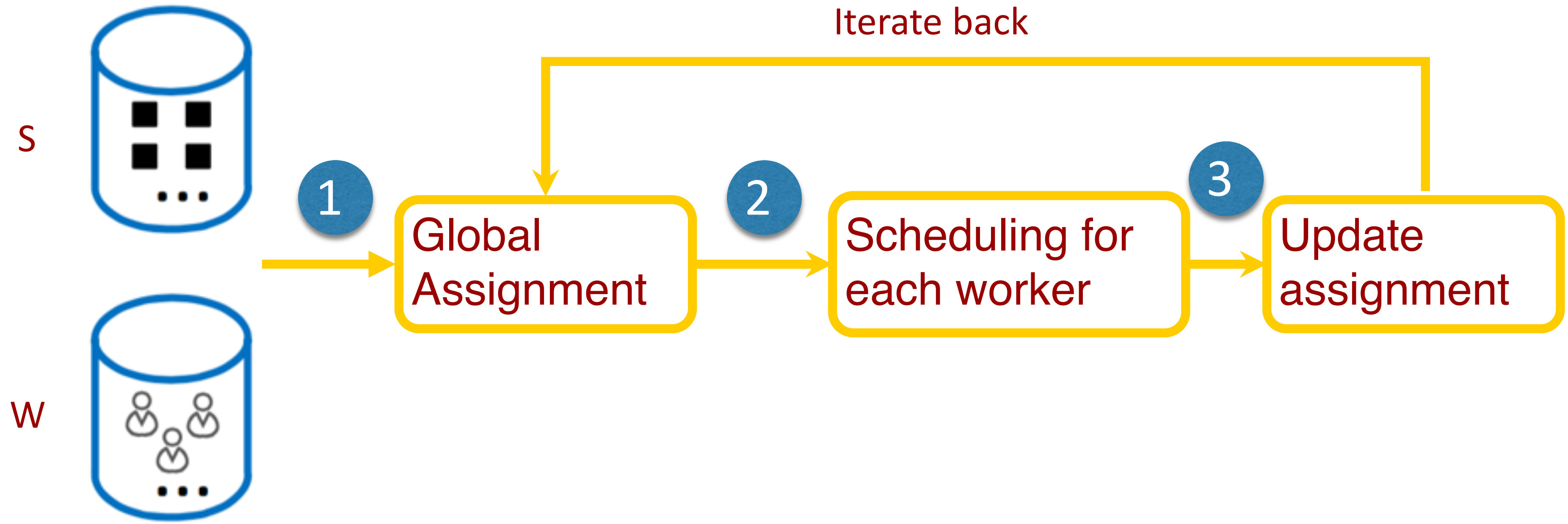
W1: S1

Reassignment and rescheduling?

W4: S6

W5: S5, S7, S8

Global assignment and local scheduling (**GALS**)



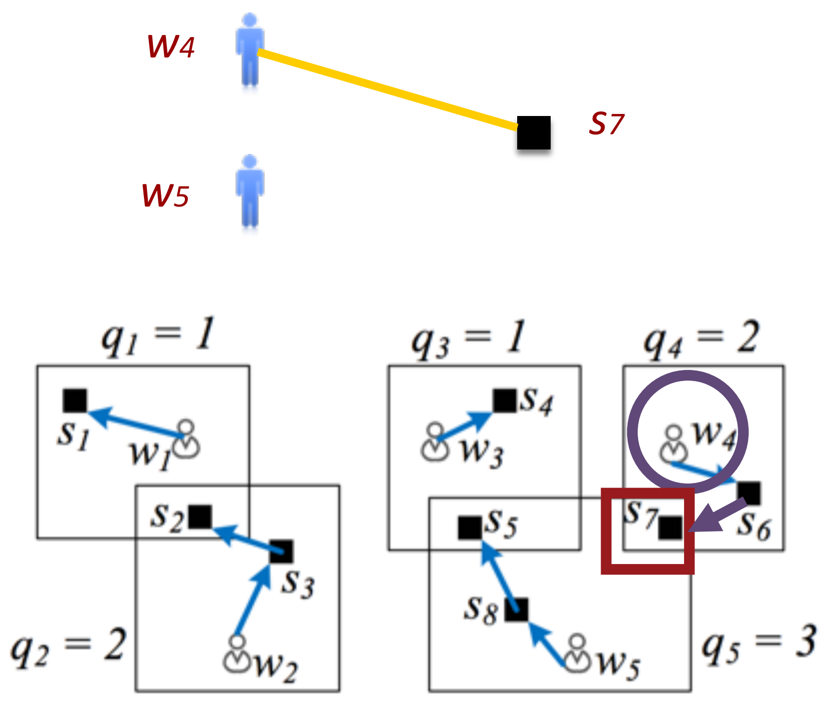
matching + scheduling **iteratively**

Example of GALS

1. Assignment via max-flow [Kazami'GIS12]

2. Schedule for each worker [Deng'GIS13]

3. Build remaining flow network and update scheduling



Insert s7 into w4's existing schedule

Property of GALS

High quality

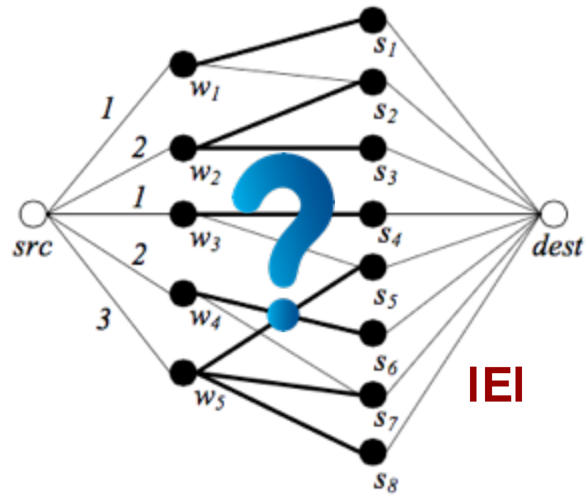
- Global assignment maintains the **connectivity** information

- The **iterative** refining process further improves the quality

50

Bottleneck of GALS

Not efficient



$O(n^2|E|)$

|E|

Suffers from the large number of edges in the flow network

For an instance with 25k tasks and 500k edges

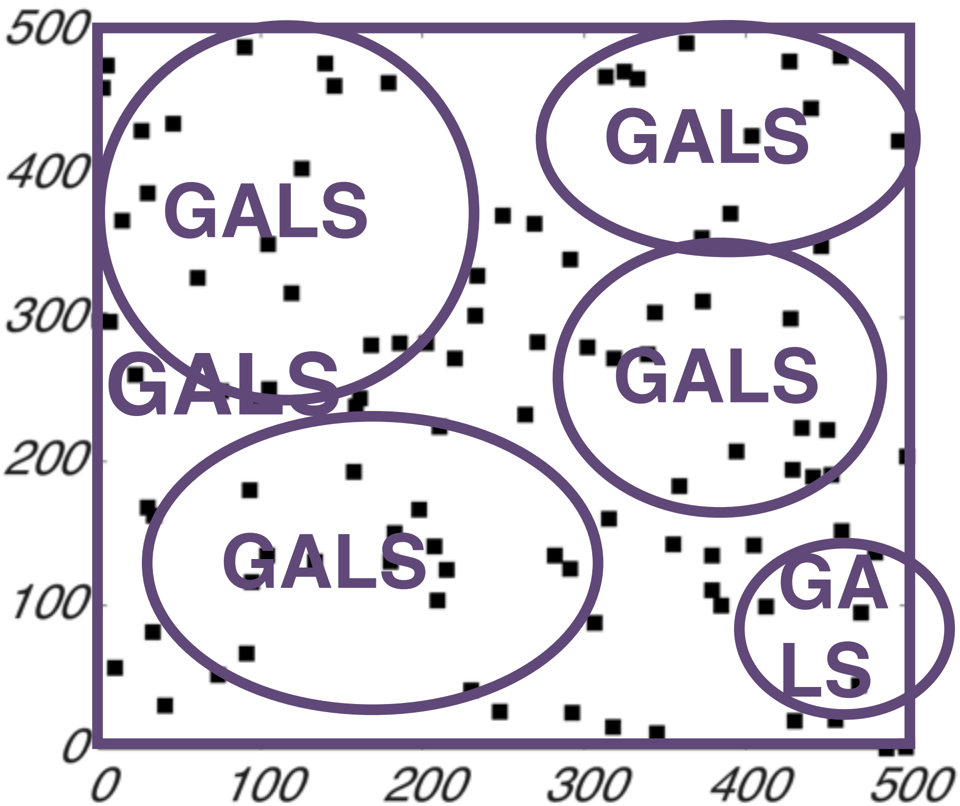
GALS takes more than **1000** seconds

Outline

- Global Assignment and Local Scheduling (GALS)
- **Local Assignment and Local Scheduling (LALS)**
  - **Naive LALS**
  - Bisection-based LALS
- Experiments

Naive LALS

Initial workers and tasks
Remaining workers and tasks



1. Generate partitions

2. Schedule for each partition GALS

3. Combine the remaining workers and tasks

Break global assignment into a set of **local assignments and local scheduling** (LALS)

Naive LALS

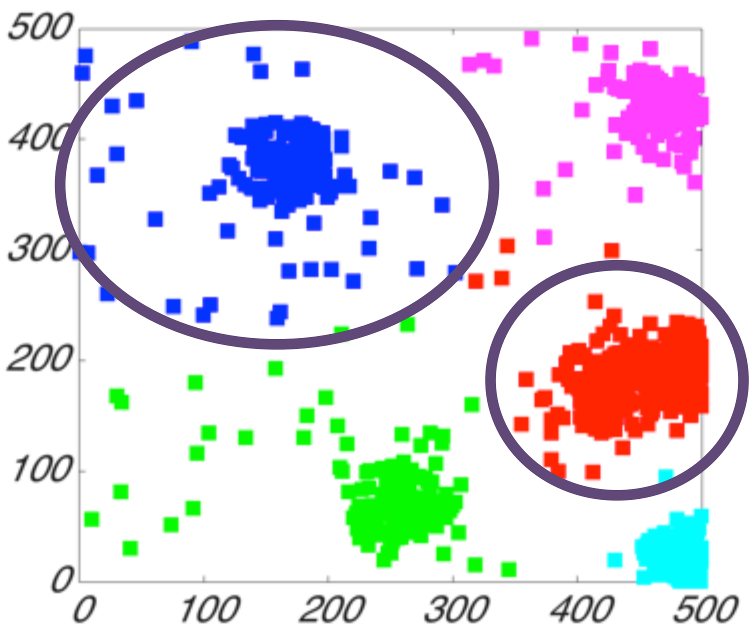
?

Faster than GALS

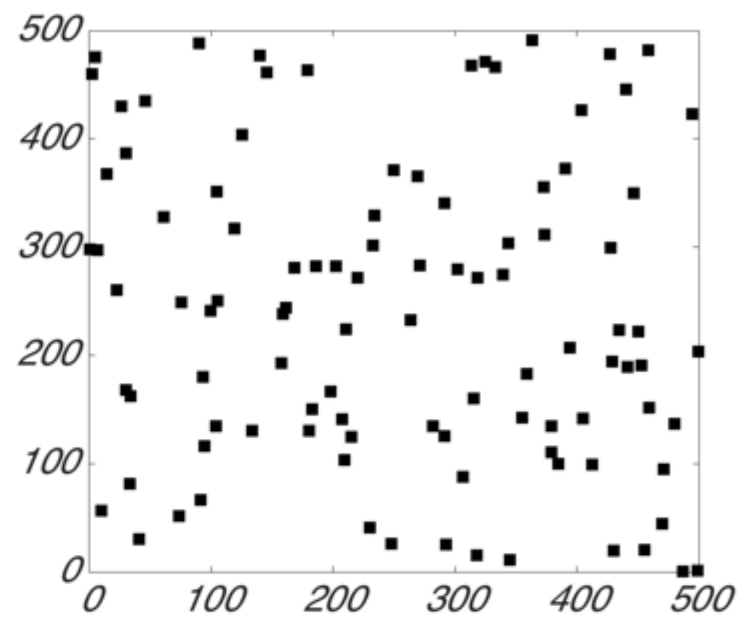
Sacrifice the solution quality

Problems

- Partitions with **large** number of edges



- **Large** remaining flow network



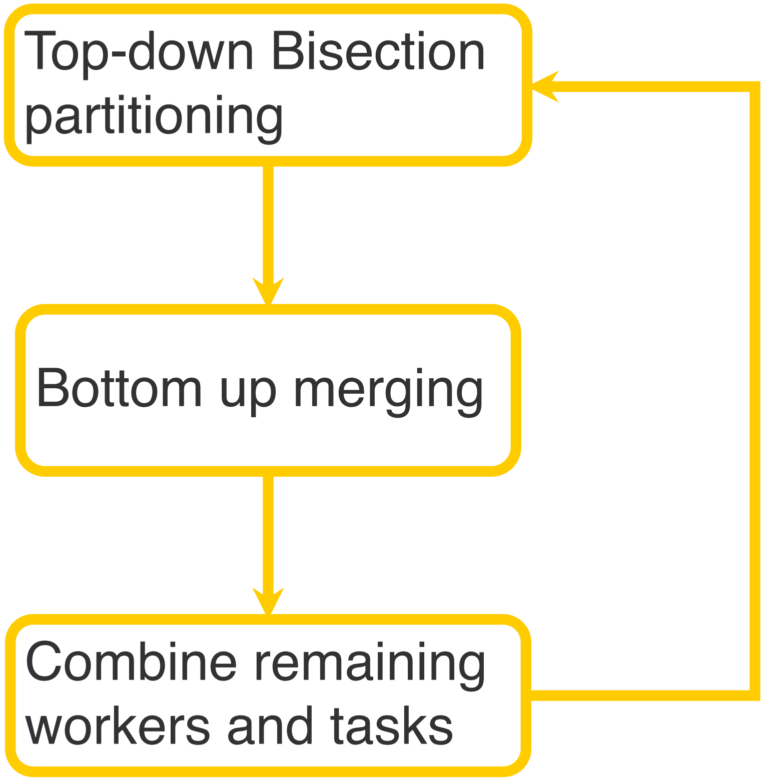
Balanced workload at each partition

Small remaining workloads

Outline

- Global Assignment and Local Scheduling (GALS)
- Local Assignment and Local Scheduling (LALS)
  - Naive LALS
  - **Bisection-based LALS**
- Experiments

Bisection-based LALS



Outline

- Global Assignment and Local Scheduling (GALS)
- Local Assignment and Local Scheduling (LALS)
  - Naive LALS
  - Bisection-based LALS
- **Experiments**

Experiment

- Dataset
  - Synthetic: **SYN-SKEW,** SYN-UNI from 500 * 500 grid
  - Real dataset from **Gowalla** and **Yelp**
- Algorithms
  - Baseline, **GALS**
  - Naive LALS (**NLALS)**, Bisection LALS(**BLALS)**

Varying |S| on SYN-SKEW

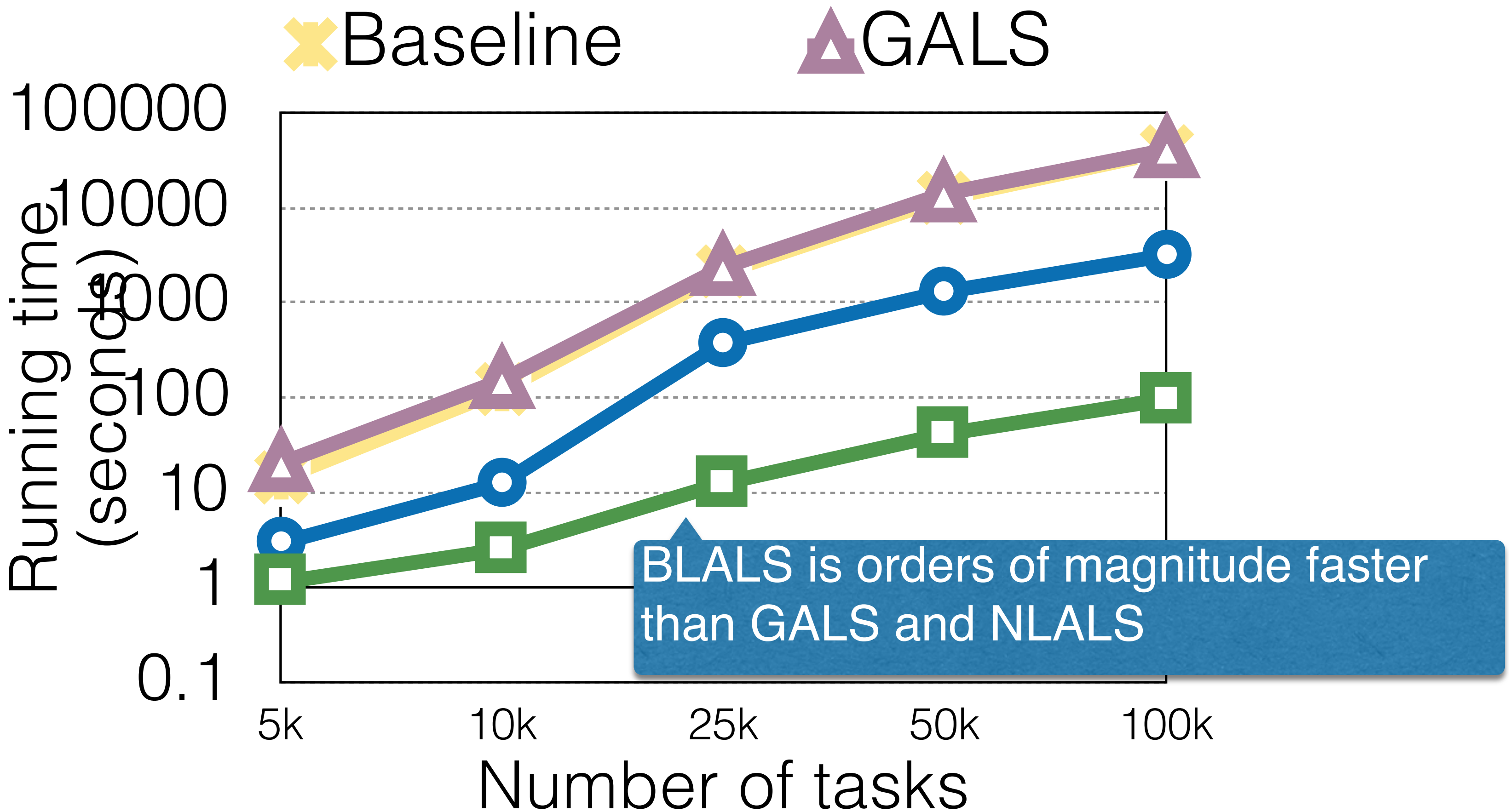
- No. of scheduled task

	baseline	GALS	NLALS	BLALS
5K	3379	**3986**	3911	3896
10K	7075	**8263**	8201	8093
25K	19049	**21849**	21717	21473
50K	35614	**43653**	43368	42095
100K	56511	**68505**	66275	63937

BLALS sacrifices only less than 5% of the schedule quality

0%

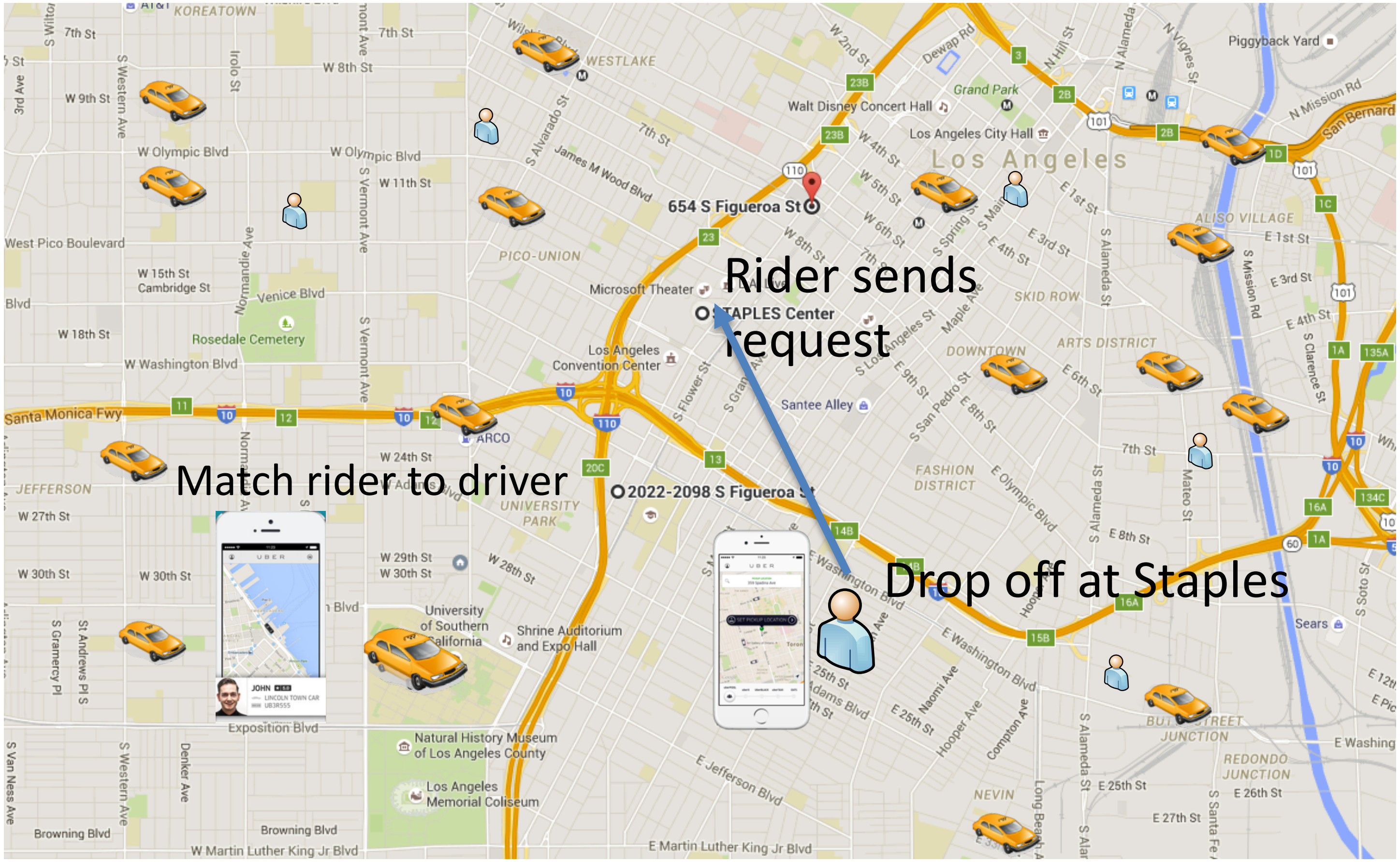
Running time on SYN-SKEW



OUTLINE

- Motivation
- Task Assignment
- Task Scheduling
- Task Assignment & Scheduling
- Example Application

Ride Sharing



Ride Sharing

Ride Sharing

Background: Dial-A-Ride Problem (OR)

- Given m vehicles at the depot and n requests with pickup and delivery time window
- Find m routes which minimizes the total routing cost
- Assumptions
  - Vehicles and request are known a priori
  - Off-line scheduling

Real-Time Ride-Matching at Scale

New Businesses

ebay now

amazon fresh

Google Shopping Express

INTRODUCING UberRUSH

FROM THE UBER GARAGE

Scale

Large nu
riders &
Large network

Dynamism

(time-dependent)

Response-Time

ne the
r and
update its route/schedule

Our Solution: Auction-based framework

Auction-Based Framework [SIGSPATIAL'16]

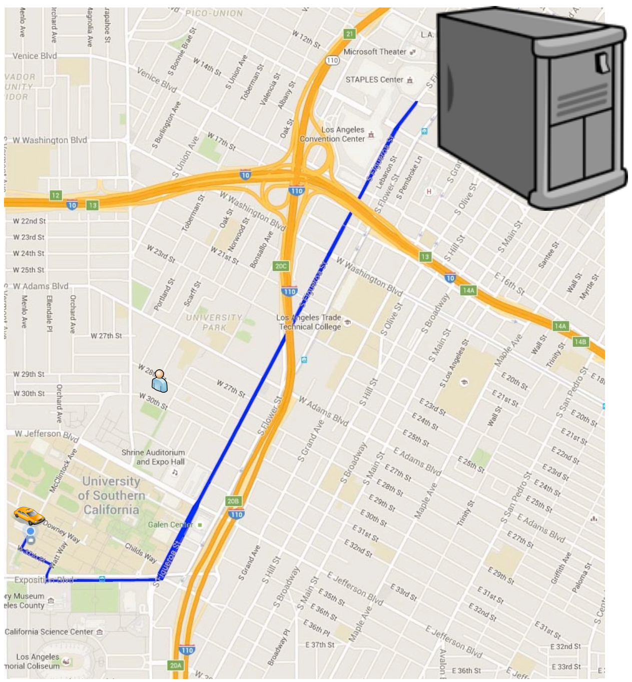
Auction-Based Framework

New ***request*** for LA Convention center

Auction-Based Framework

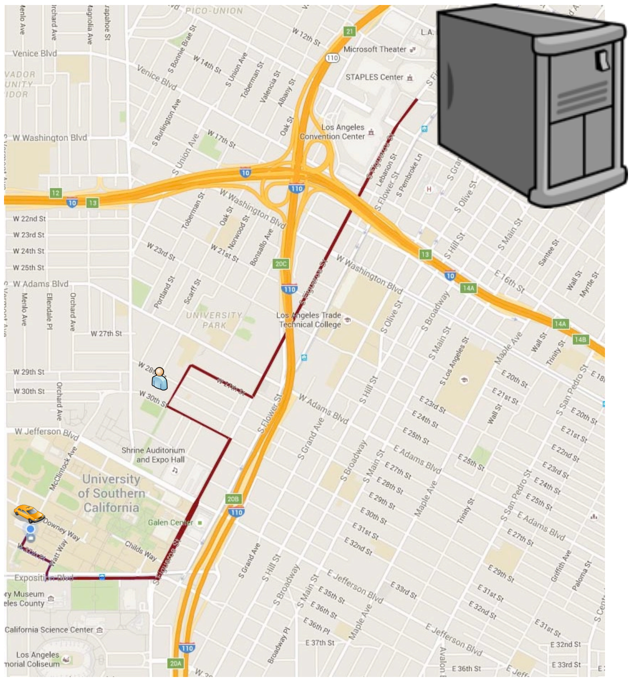
Send request to ***nearby*** drivers

Auction-Based Framework



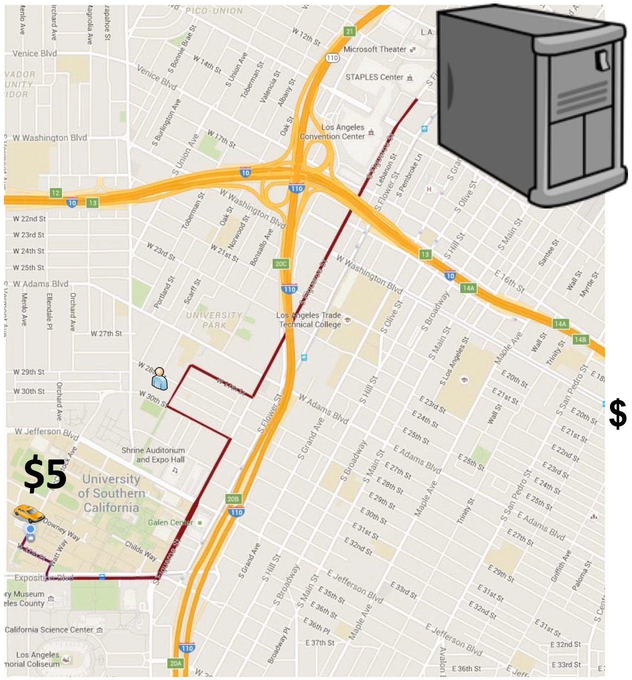
- Each driver has a ***current*** schedule

Auction-Based Framework

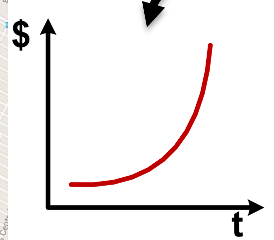


- Each driver has a ***current*** schedule
- Each driver computes a ***best potential*** schedule
- ***detour*** = *diff(**best potential**, **current**)*

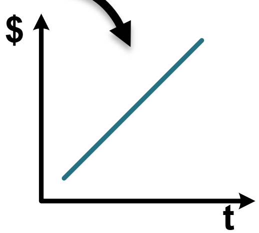
Auction-Based Framework



- Each driver has a ***current*** schedule
- Each driver computes a ***best potential*** schedule
- ***detour*** = *diff(**best potential**, **current**)*
- ***bid*** = *profit(**request, detour, user_networth, driver_fare**)*

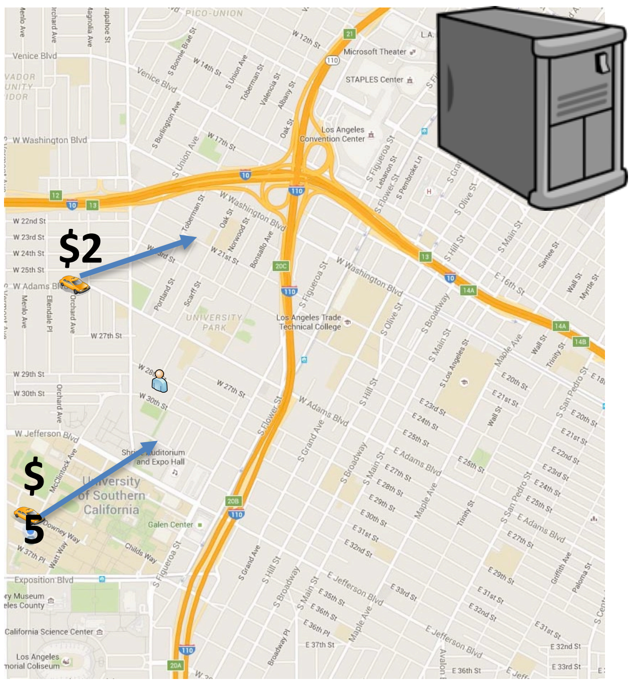


User Networth



Driver Fare

Auction-Based Framework



A bid can be thought of as the “profit for Uber to add this ride”

Server receives bids from nearby drivers and assigns request to ***highest*** bidder.

Real-Time Ride-Matching at Scale

Our Solution: Auction-based framework

Scale

1. Local scheduling of a small number of riders per driver
2. Simple ranking across bids by the server

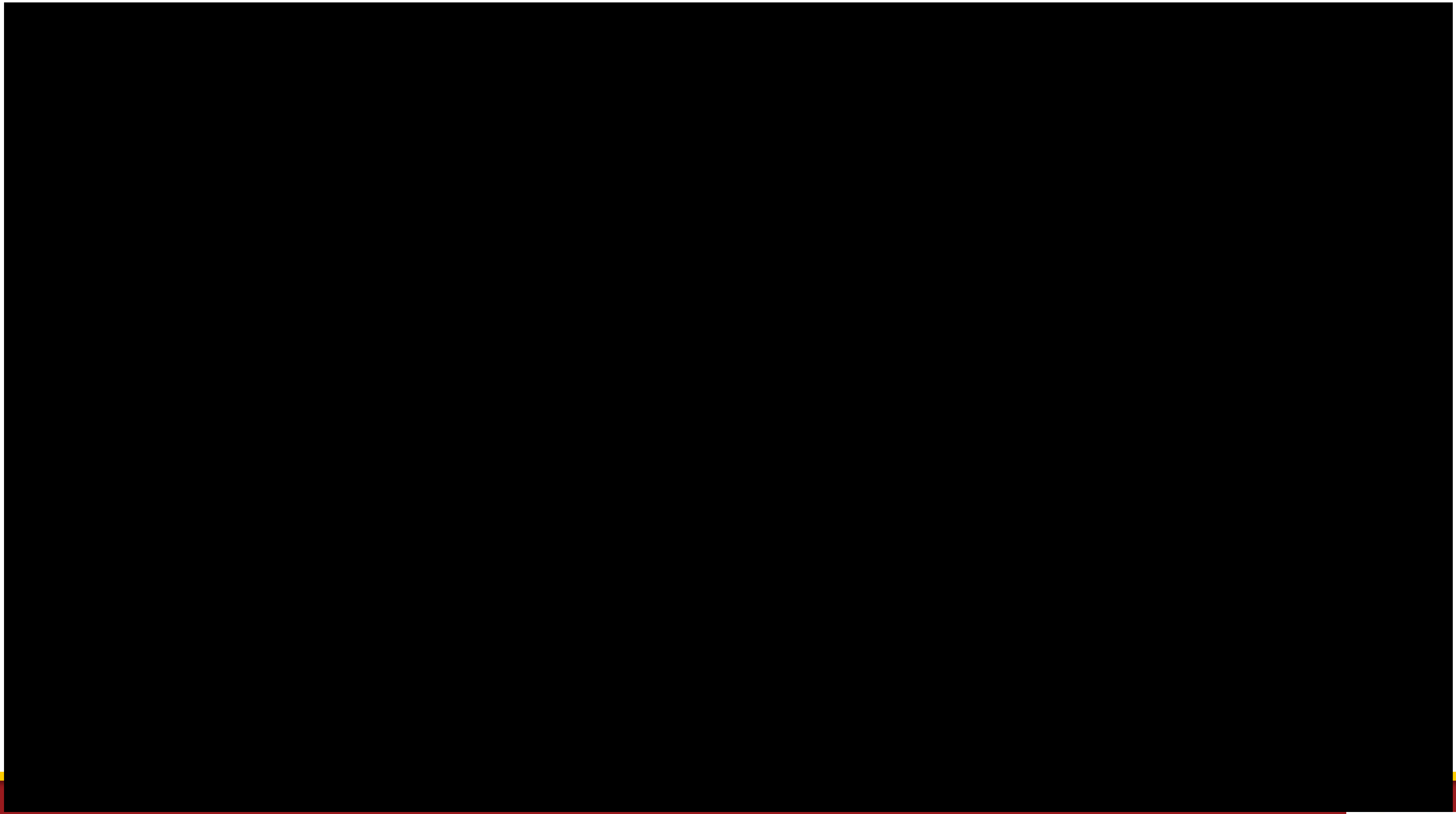
Dynamism

1. Bidding is triggered per rider's arrival
2. Local time-dependent routing per driver

Response-Time

In order of milliseconds (<300ms, interactive) per our preliminary experiments http://mediaq.usc.edu/

MediaQ Demo



Research Challenges in Spatial Crowdsourcing

Task assignment (or worker selection)

Process of identifying which tasks should be assigned to which workers

- **Asghari et al. SIGSPATIAL 2016**
- Bessai and Charoy ISCRAM '16
- Hassan and Curry ESA'16
- Zhang et al. TVT '16
- Gao et al. WAIM '16
- Cheng et al. TKDE '16
- Tong et al. VLDB '16
- Liu et al. DASFAA '16
- Hu et al. ICDE '16
- Tong et al. ICDE'16
- Zhang et al. WCMC '16
- Liu et al. UbiComp '16
- Guo et al. THMS '16
- **To et al. PerCom '16**
- **To et al. TSAS '15**
- **Alfarrarjeh et al. MDM '15**
- Fonteles et al. MoMM '15
- Hassan and Curry. SIGSPATIAL '15
- Xiao et al. INFOCOM '15
- Xiong et al. PerCom '15
- Pournajaf et al. ICCS '14
- Hassan and Curry. UCI'14
- He et al. INFOCOM '14
- Fonteles et al. SIGSPATIAL '14
- Zhang et al. UbiComp '14
- **Dang et al. iiWAS '13**
- **Kazemi and Shahabi. SIGSPATIAL '12**

Research Challenges in Spatial Crowdsourcing

Privacy-preserving task assignment

- **To et al. TMC ’16**
- Zhang et al. CN ‘16
- Zhang et al. ATIS ‘15
- Shen et al. GLOBECOM ‘15
- Gong et al. IoT’15
- Gong et al. TETC’15
- Hu et al. APWeb ’15
- Pournajaf et al. MDM’14, SIGSPATIAL’15
- **To et al. VLDB ’14, ICDE ’15**
- Boutsis and Kalogeraki PerCom ‘13
- Vu et al. INFOCOM ‘12
- **Kazemi and Shahabi SIGKDD ’11**

Task scheduling

Path planning for workers to perform tasks

- Wang et al. 2016
- Fonteles et al. JLBS ‘16
- **Deng et al. GeoInformatica ’16**
- Mrazovic et al. ICDMW ‘15
- Chen et al. IJCAI ‘15
- Chen et al. AAMAS ‘15
- Hadano et al. HCOMP ’15
- **Deng et al. SIGSPATIAL ’15**
- Chen et al. HCOMP ‘14
- **Deng and Shahabi. SIGSPATIAL ’13**

Research Challenges in Spatial Crowdsourcing

Trust and Quality

Consider quality of the report data or trustworthiness of workers

- Liu et al. Sensor ’16
- Zhang et al. TETC ’16
- Miao et al. DSS ’16
- Fan et al. SOSE ’15
- Shah-Mansouri et al. ICC ’15
- An et al. HPCC ’15
- Kang et al. MASS ’15
- Cheng et al. VLDB ’15
- Zhao et al. MDM ’15
- Wang et al. UbiComp ’15
- Song et al. TVT ’14
- Boutsis et al. ICDCS ’14
- Feng et al. INFOCOM ’14
- **Kazemi et al. SIGSPATIAL ’13**

Incentive mechanism

Incentivize workers to perform spatial tasks

- Zhang et al. TVT ’16
- Kandappu et al. CSCW ‘16
- Kandappu et al. UbiComp ‘16
- Micholia et al. IJHCS ’16
- **To et al. GeoRich ’16**
- Li and Cao TMC ‘16
- Thebault-Spieker et al. CSCW ’15
- Jin et al. MobiHoc ’15
- Teodoro et al. CSCW ’14
- Rula et al. HotMobile ‘14
- Musthag et al. CHI ’13
- Heimerl et al. CHI ’12
- Jainmes et al. PerCom ‘12
- Yang et al. MobiCom ‘12
- Lee and Hoh PMC ‘10
- Alt et al. NordiCHI ‘10

Research Challenges in Spatial Crowdsourcing

Generic frameworks

Discuss components, architecture, programming framework of SC apps

- **To et al. CROWDBENCH ’16**
- Fonteles et al. RCIS ‘16
- Peng et al. ASE ’16
- Kucherbaev et al. SIGCHI ’16
- Sakamoto et al. COMPSAC ’16
- Fernando et al. MOBIQUITOUS ’13
- Tamilin et al. UbiComp ‘12
- Ra et al. MobiSys ‘12
- Yan et al. SenSys ‘09

Related surveys

- Pournajaf et al. SIGMOD ‘15
- Guo et al. Comp Survey ‘15
- Zhao and Han 2016
- Christin JSS ’15

Applications

- Konomi and Sasao Urb-IoT ‘16
- Jaiman et al. UbiComp/ISWC ’16
- Fan and Tseng MOBIS ‘15
- Konomi and Sasao UbiComp/ISWC ’15
- Harburg et al. CHI ’15
- Chen et al. SenSys ‘15
- Kim CHI ‘15
- Aubry et al. CROWDSENSING ’14
- Chen et al. VLDB ’14
- **Kim et al. MMSys’14**
- Benouaret et al. IEEE IC ’13
- Coric and Gruteser DCOSS ’13
- Koukoumidis et al. MobiSys ‘11
- Goodchild and Glennon IJDE ‘10