Efficient Mobile Crowdsourcing via Gamification for Smart City Applications

在智慧城市中遊戲化手機群眾外包系統

Ying-Yi Chen
Advisor: Cheng-Hsin Hsu
Networking and Multimedia Systems Lab
CS Dept. ,National Tsing Hua University

Outline

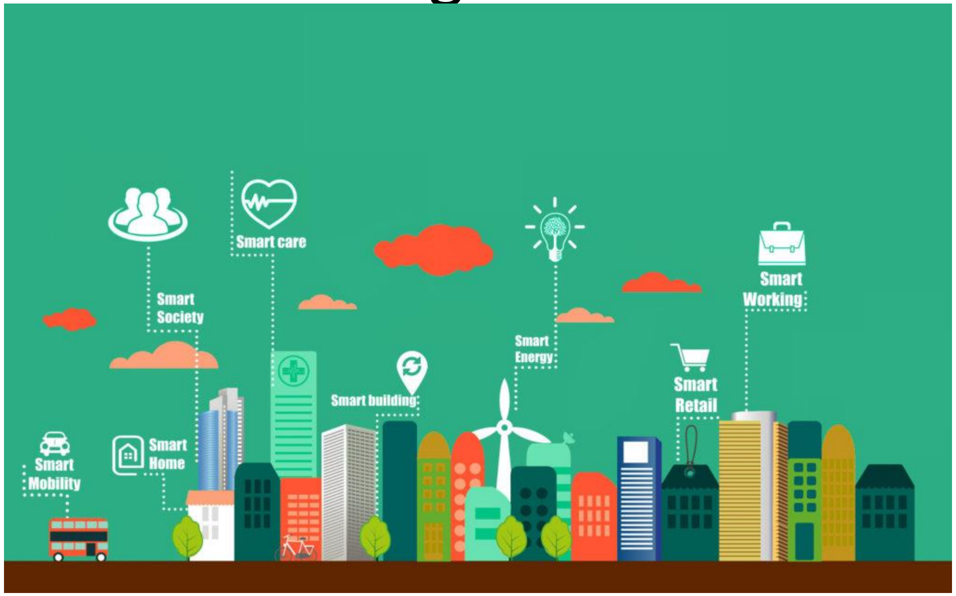
- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

Outline

- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

Motivation

- Smart city market will grow at an annual rate of almost 20% and reach 1.45 trillion USD by 2020
- Smart cities require intelligent infrastructures to solve various resource management problems and large-scale social/economic challenges
  - Traffic congestion
  - Pollution monitoring
  - Energy consumption



Mobile Crowdsourcing

- Mobile crowdsourcing enhancing infrastructure sensing.
- A hybrid sensing platform with in-situ sensors and smartphones, called Smartphone Augmented Infrastructure Sensing (SAIS) [1]
- Dispatch mobile users to the right place at the right time for performing the sensing tasks. A crowdsourcing platform can help researchers collect data
    - Noise
    - PM 2.5
    - Road conditions
    - Picture、Video

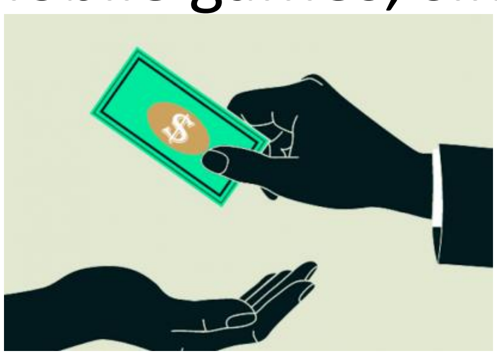
[1] C. Liao, T. Hou, T. Lin, Y. Cheng, A. Erbad, C. Hsu, and N. Venkatasubramania. Sais: Smartphone augmented infrastructure sensing for public safety and sustainability in smart cities. In Proceedings of the International Workshop on Emerging 36 Multimedia Applications and Services for Smart Cities (EMASC'14), pages 3–8, Orlando, FL, November 2014.

Outline

- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

Incentive Mechanism

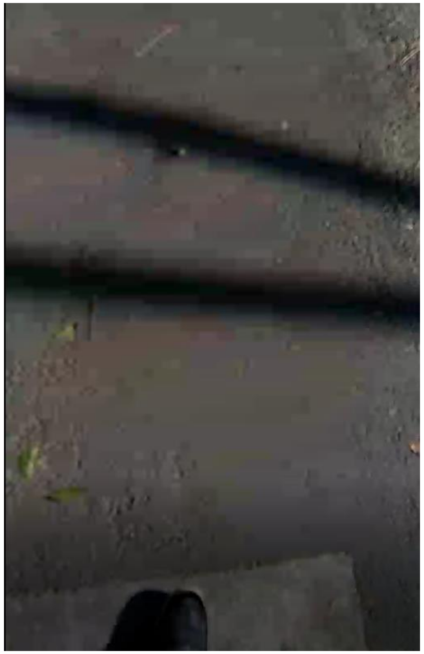
- Performing sensing tasks is energy and time consuming
- Monetary incentives
  - Determining the price of each crowdsourced sensing task is a difficult problem
- We propose to gamify the SAIS platform using mobile games, similar to the popular Pokémon Go [2]



[2] Y. Chen, H. Hong, S. Yao, A. Khunvaranont, and C. Hsu. Gamifying Mobile Applications for Smartphone Augmented Infrastructure Sensing. In Proceedings of the 15th Annual Workshop on Network and Systems Support for Games (NetGames'17). June , 2017.

Task Coverage

- Some task need directional sensing, such as taking photos or shooting videos
- We may need to assign more than one gamer to cover 360 degrees

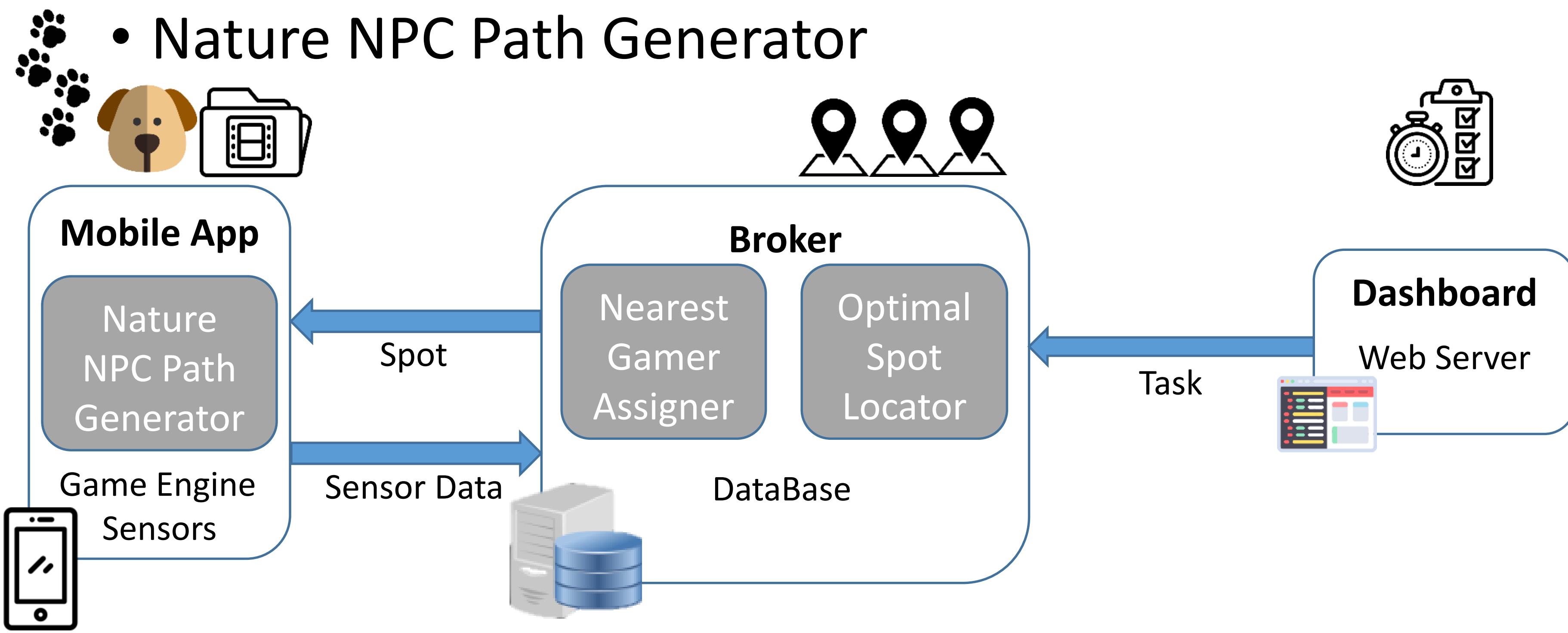


Outline

- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

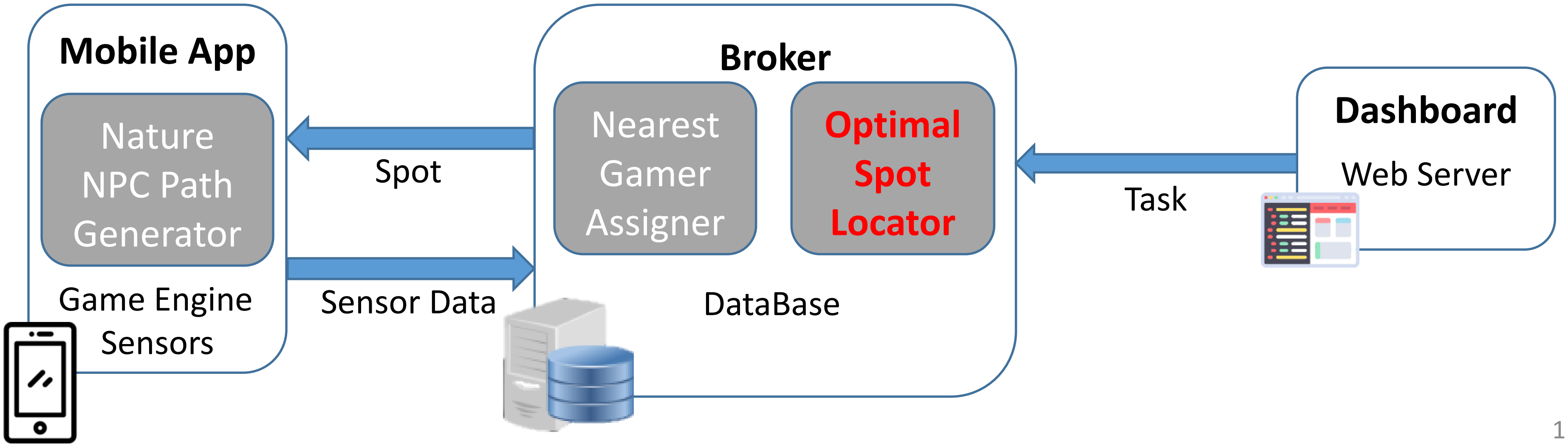
Problems & Solutions

- There are three problems we solved in our system
- Optimal Spot Locator
- Nearest Gamer Assigner
- Nature NPC Path Generator



Optimal Spot Locator

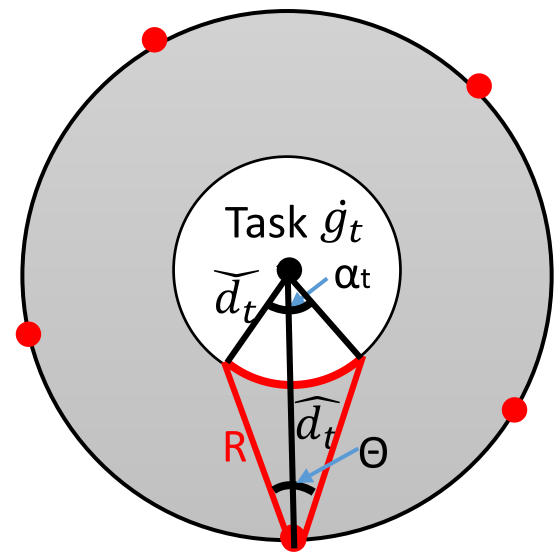
- To locate the least number of spots to finish a task



Algorithm-Optimal Spot Locator

Optimal Spot Locator

- T be the number of sensing tasks
- U_t be the requested angle set of task t
- maximal effective distance $\hat{d}_t$
- minimal effective distance $\check{d}_t$



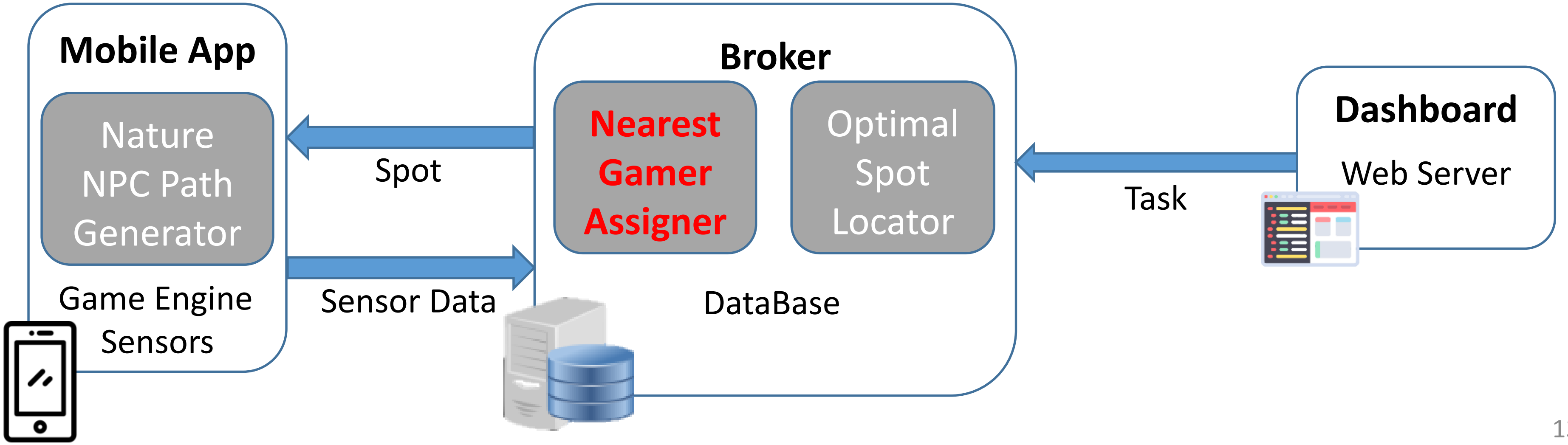
Algorithm 1 Optimal Spot Locator

```
function SPOT_LOCATOR
    for each task t = 1, 2, . . . , T do
        for e ∈ U_t do
            N = ⌈(max(e) − min(e))/α_t⌉
            for n = 0, 1, . . . , N − 1 do
                S_t = S_t ∪ {((max(e) − min(e))/N)n + min(e)}
```

$$\alpha_t = 2arccos((\hat{d}_t^2 + \check{d}_t^2 - R^2)/2\hat{d}_t\check{d}_t)$$

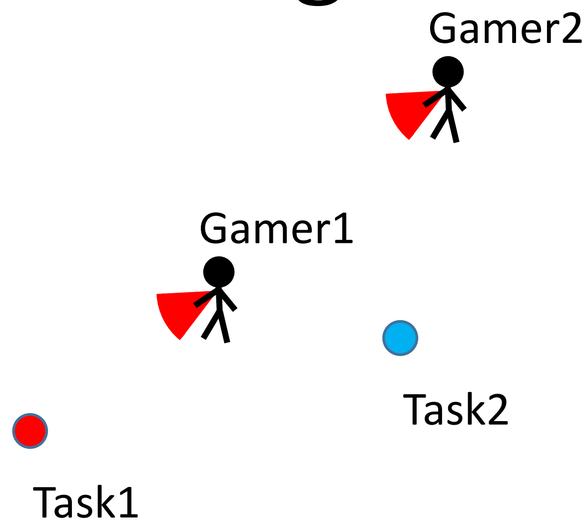
Nearest Gamer Assigner

- Assigned gamer must have the sensors required by the task
- Assigns gamers to nearest tasks
- Periodically executed, say once every 5 minutes



Algorithms-Nearest Gamer Assigner

- g_p be the current location of gamer p
- $\dot{g}_t$ be the GPS location of task t
- $\hat{l}_t$ be the beginning life time of task t
- $\check{l}_t$ be the end life time of task t

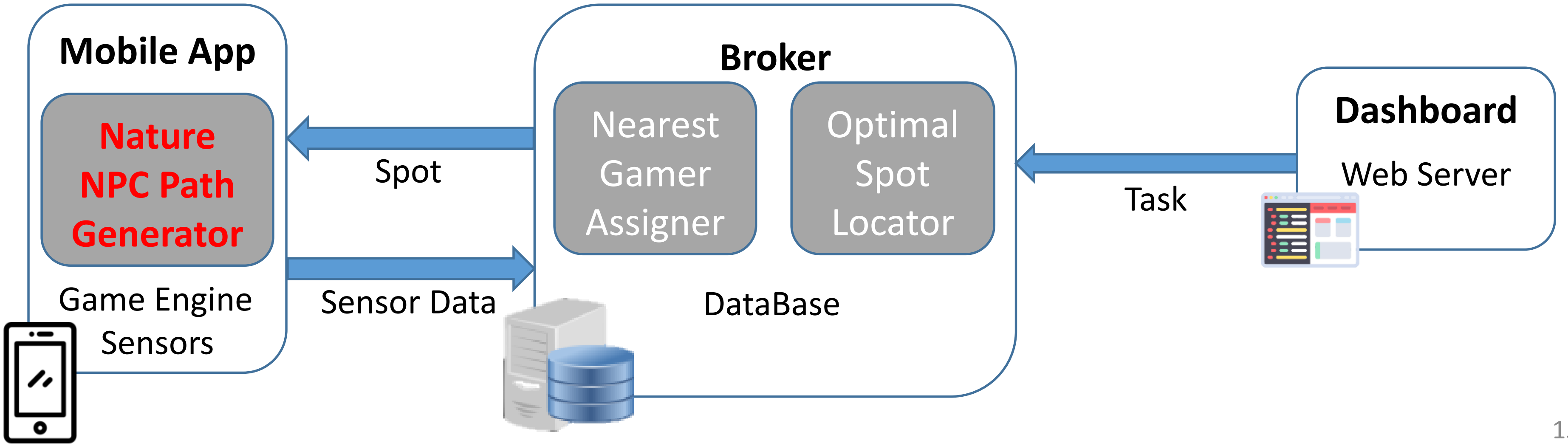


Algorithm 2 Nearest Gamer Assigner

function GAMER_ASSIGNER
  while there are remaining tasks **do**
    find the most urgent task $\dot{t}$ by comparing $\check{l}_t$ and current time
    while there are unsatisfied spots of $\dot{t}$ **do**
      Find the nearest $\dot{p}$ by comparing $g_{\dot{t}}$ and g_p
      if gamer $\dot{p}$ has required sensors for task $\dot{t}$ **then**
        assign $\dot{p}$ to the nearest unsatisfied spot $\in \mathbf{S}_{\dot{t}}$ in $\mathbf{Q}_{\dot{t}}$

Nature NPC Path Generator

- Creates the NPC paths to guide the gamers
- In order to make the gamers recording the video at the right place with the right angle

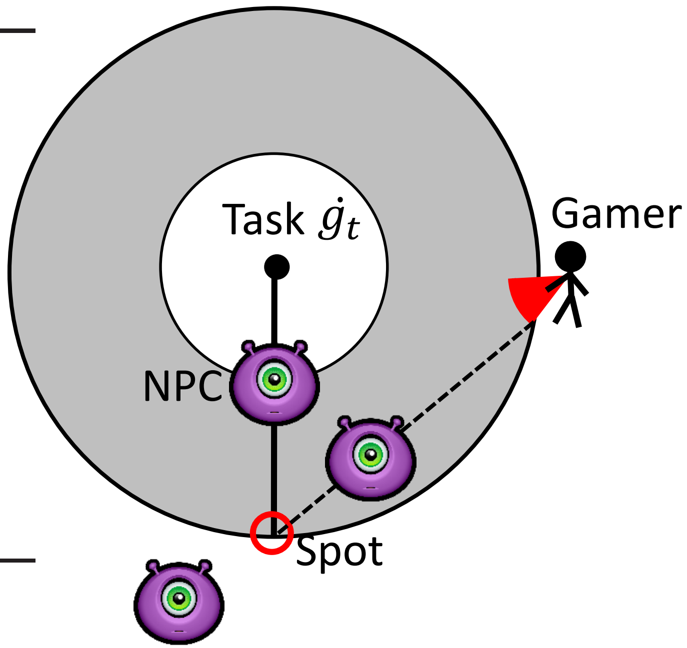


Algorithm-Nature NPC Path Generator

- g_{ts} be the location of spot s and task t

Algorithm 3 Nature NPC Path Generator

function PATH_GENERATOR
  Let gi be the initial GPS location of NPC
  $gi = \dot{g}_t + (\overrightarrow{\dot{g}_t g_{ts}})/\hat{d}_t \check{d}_t$
  Put NPC at the initial point $Gn = g_i$
  while g_p is not at spot location g_{ts} **do**
    $Gn = g_p + (\overrightarrow{g_p g_{ts}})/\hat{d}_t(\hat{d}_t - \check{d}_t)$
  while Gn is not at gi **do**
    $Gn = Gn + (\overrightarrow{Gngi})/(|\overrightarrow{Gngi}|)$



Outline

- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

Simulation settings

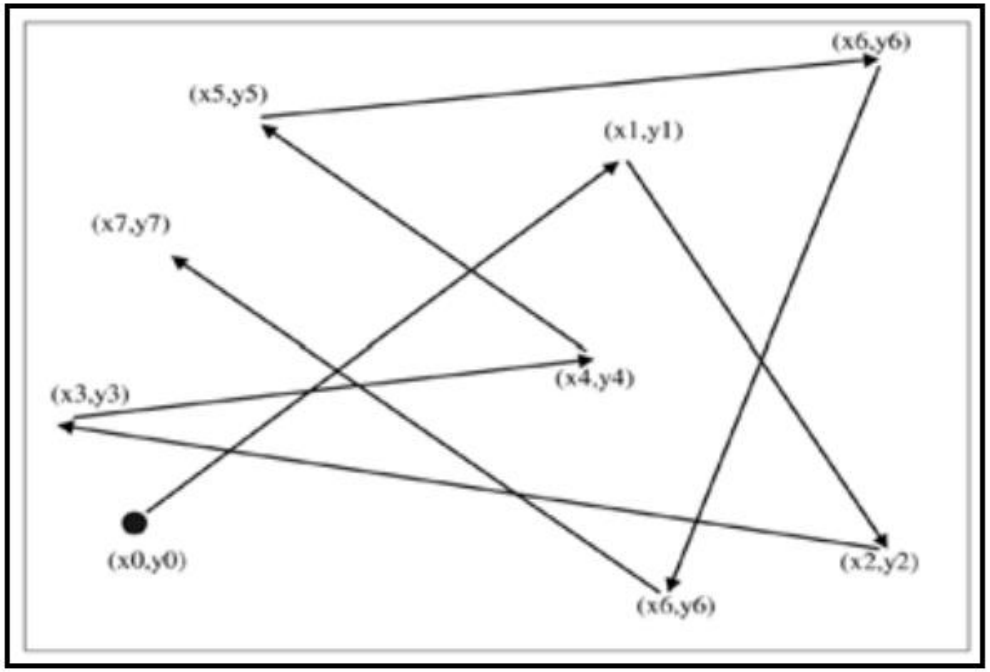
- 5*5 km^2
- Each gamer has 1 to 3 hours available between 6 a.m. and 6p.m. every day
- The gamers move around until they are assigned a task
- Travel speed of gamers is 5.4 km/hr
- Time: 7 days
- Number of gamers P = {25, 50, **100**, 200, 400}
- Number of tasks T = {100, **200**, 400, 800}
- Life time of tasks l_t = {1, 2, **3**, 4, 5} (hr)

Mobility Models

- Random Waypoint Model
- Pathway Mobility Model

Random Waypoint Model

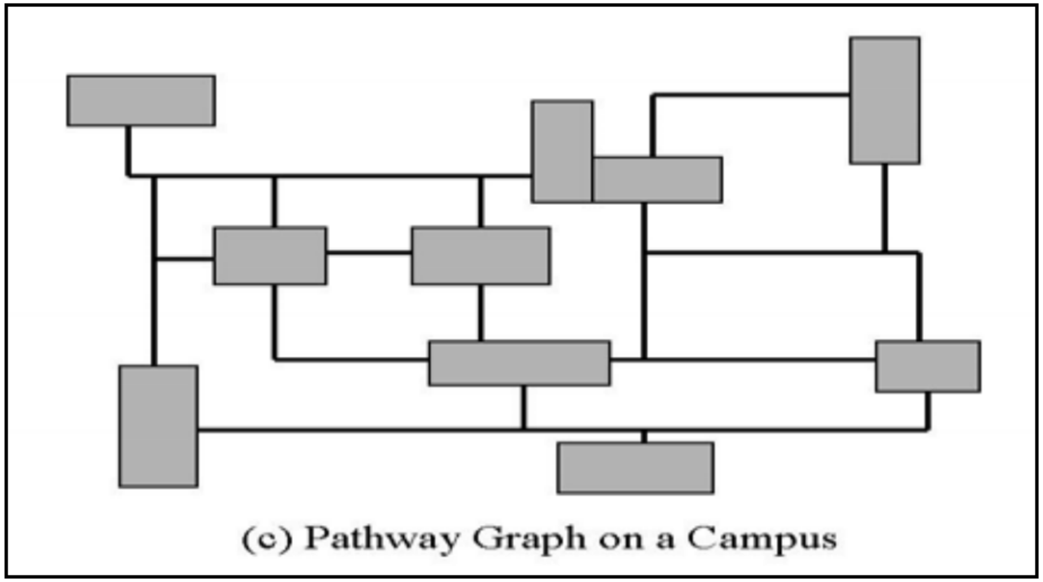
- Each gamers moves independently to a randomly chosen destination in the simulation area.



Source: F. Bai and A. Helmy. A survey of mobility models.
Wireless Adhoc Networks. University of Southern California, USA, 206:147, 2004.

Pathway Mobility Model

- To restrict the gamers' movement to the pathways in the map.
- We extract the roads from OpenStreetMap.



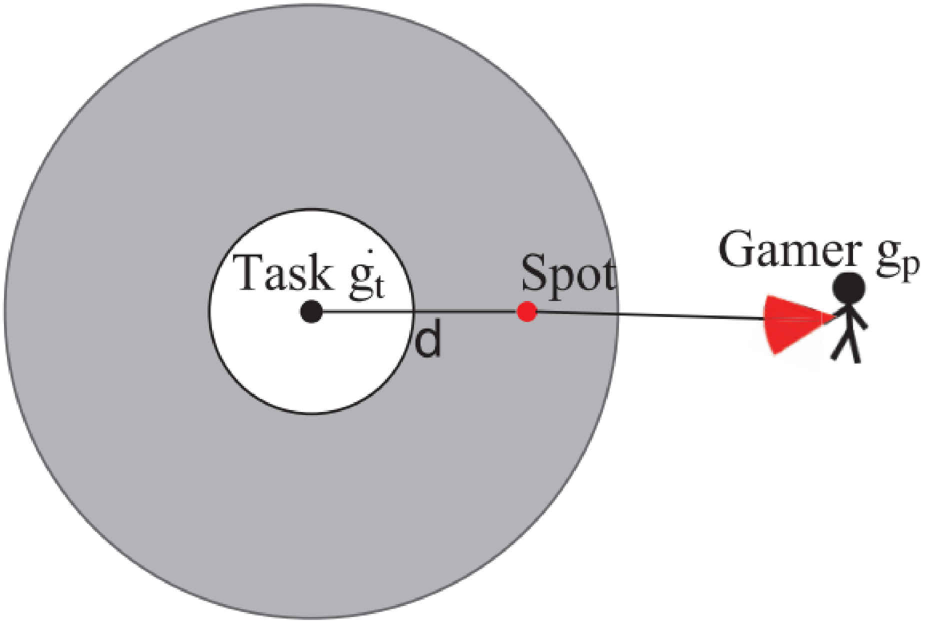
Source: F. Bai and A. Helmy. A survey of mobility models.
Wireless Adhoc Networks. University of Southern California, USA, 206:147, 2004.

Baseline Algorithms

- Current Practice (CP): mimics manual assignments and human behaviors
- Video Surveillance Networks (VSN) [3]: randomly selects the spots within the effective

[3] T. Chen, H. Tsai, C. Chen, and J. Peng. Object coverage with camera rotation in visual sensor networks. In Proceedings of the 6th international wireless communications and mobile computing conference (IWCMC'10), pages 79–83, France, 2010.

Current Practice

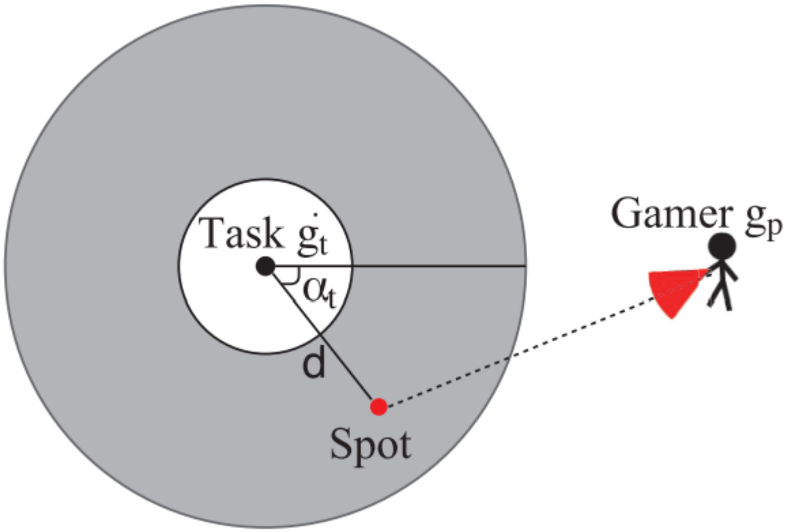


Algorithm 5 The Current Practice algorithm

```
function CURRENT_WORK
    for each task t = 1, 2, ..., T do
        for p who is assign to t do
            let d = distance away from ġ_t, which is randomly chosen between d̂_t and ď_t followed by gaussian distribution
            S_t = S_t ∪ {(ġ_t + (→ġ_t g_p)/|→ġ_t g_p| d)}
```

- **function** CURRENT_WORK
  - **for** each task $t = 1, 2, \ldots, T$ **do**
    - **for** p who is assign to t **do**
      - let d = distance away from $\dot{g}_t$, which is randomly chosen between $\hat{d}_t$ and $\check{d}_t$ followed by gaussian distribution
      - $\mathbf{S_t} = \mathbf{S_t} \cup \{(\dot{g}_t + (\overrightarrow{\dot{g}_t g_p})/|\overrightarrow{\dot{g}_t g_p}|d)\}$

Video Surveillance Networks



Algorithm 4 The VSN algorithm.

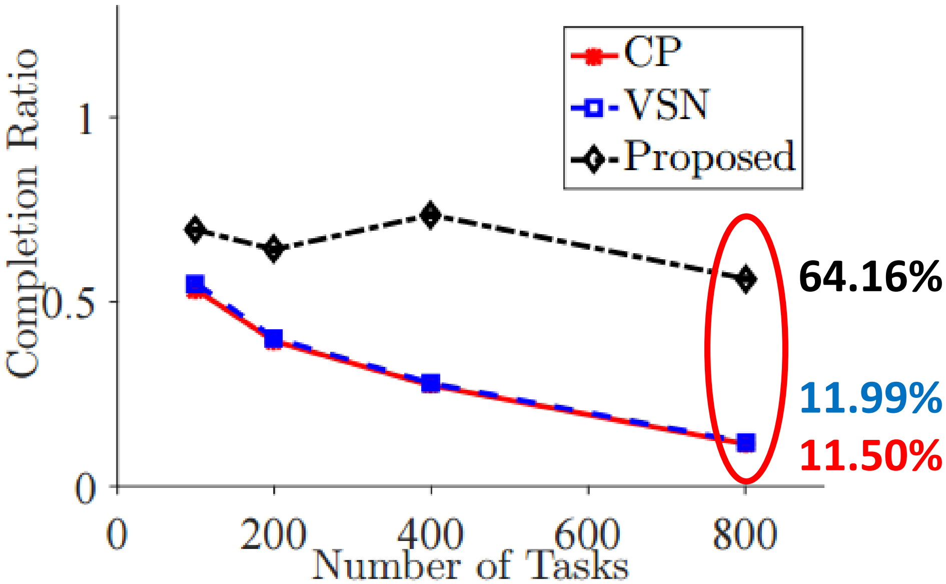
function VSN
  for each task $t = 1, 2, \ldots, T$ **do**
    for p who is assign to t **do**
      let d = distance away from $\dot{g}_t$, which is randomly chosen between $\hat{d}_t$ and $\check{d}_t$ followed by gaussian distribution
      let α_t = cover angle of task t, which is randomly chosen between 0 and 2π
      $\mathbf{S_t} = \mathbf{S_t} \cup \{\dot{g}_t + d|\sin\alpha_t, \cos\alpha_t|\}$

Performance Metrics

- Completion ratio
  - For each task, the completion ratio is the percentage of the requested angles covered by the gamers. For all tasks, the completion ratio is the average ratio across all tasks
- Response time
  - The time between the task received (from smart city applications) and completed (by gamers)
- Working hour
  - The average hours spent by gamers when carrying out the tasks
- Spots per task
  - The number of resulting spots for each task

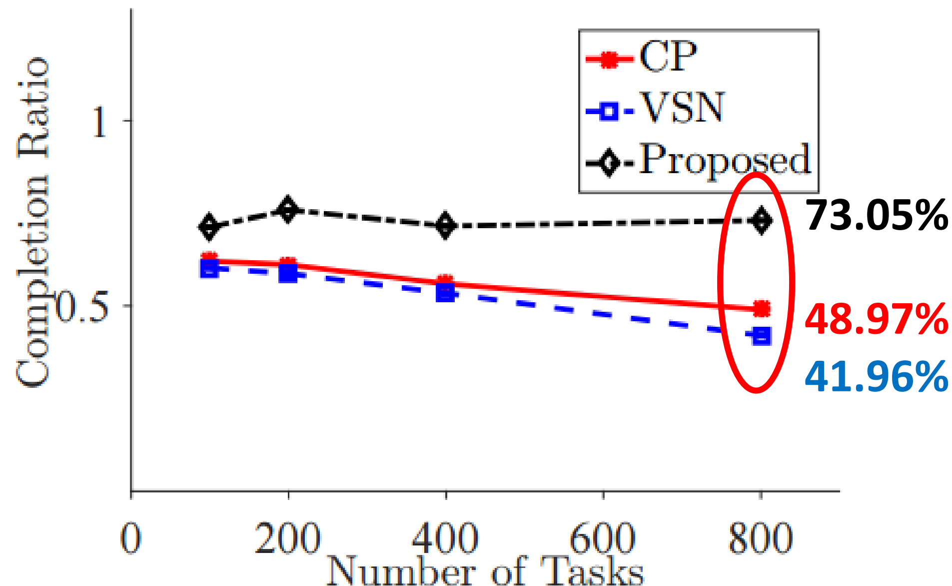
Completion Ratio

Random Way Point



Our algorithm outperforms the baseline algorithms by **4.8** times at most

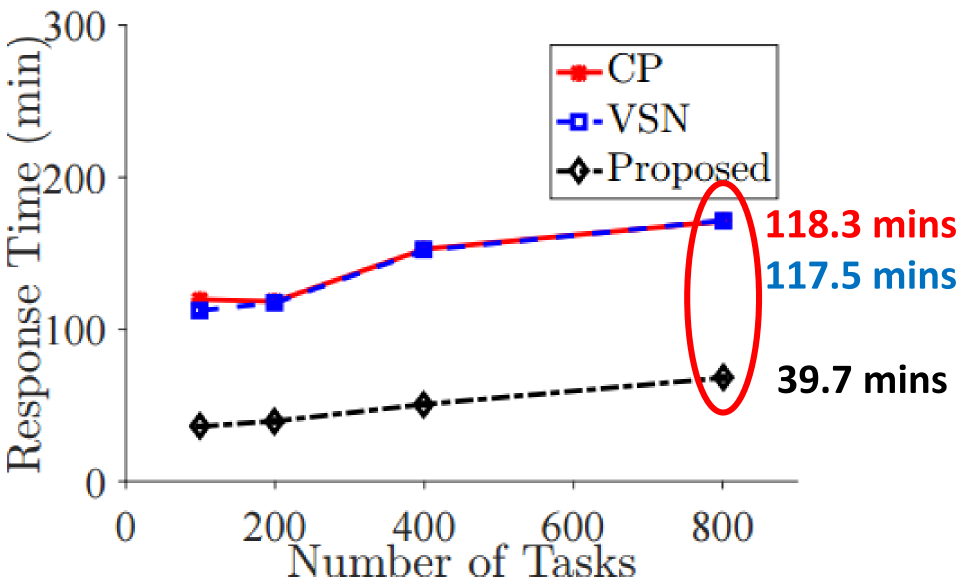
Pathway Mobility Model



Our algorithm outperforms the baseline algorithms by up to **74%**

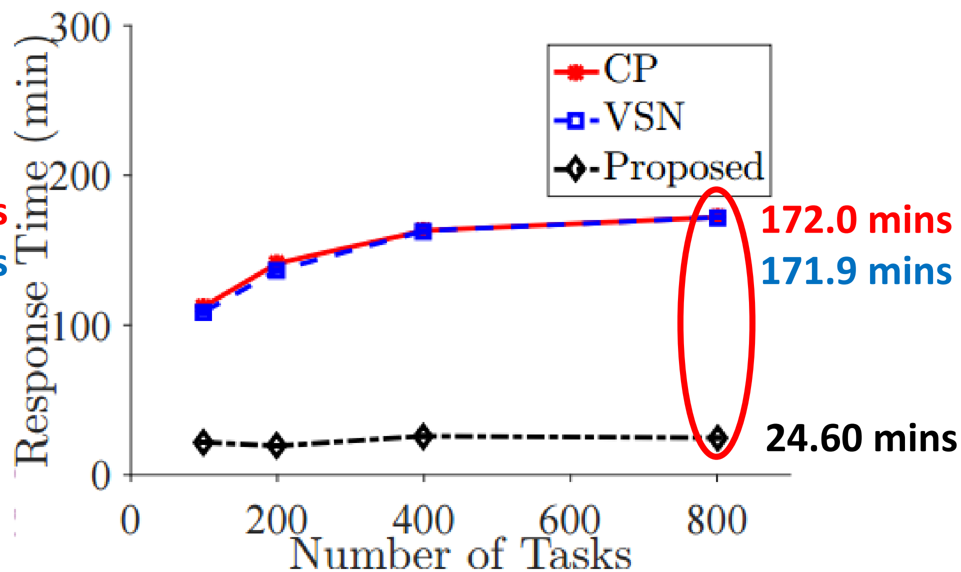
Response Time

Random Way Point



Response time of our algorithm is **less than half** of that from the baseline algorithms

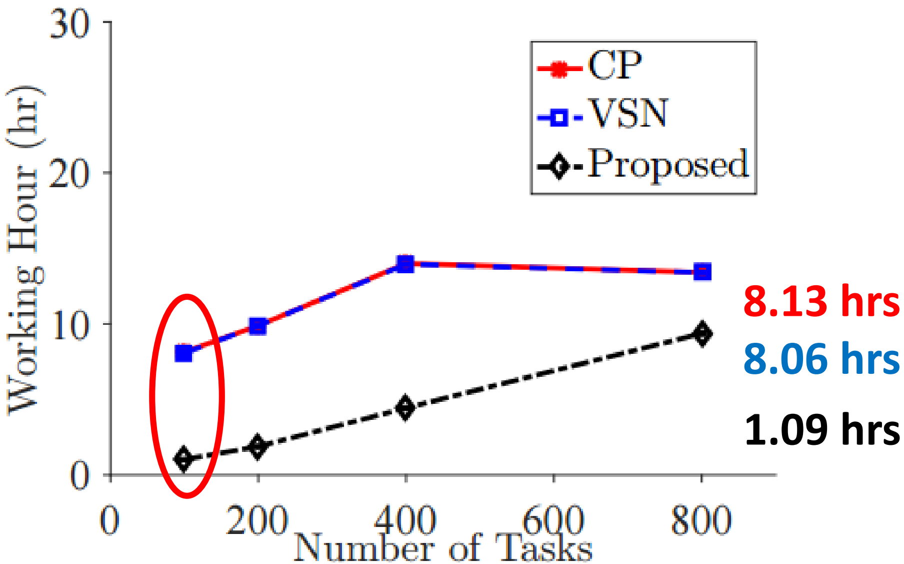
Pathway Mobility Model



The baseline algorithms have almost **7** times response time compared to our solution

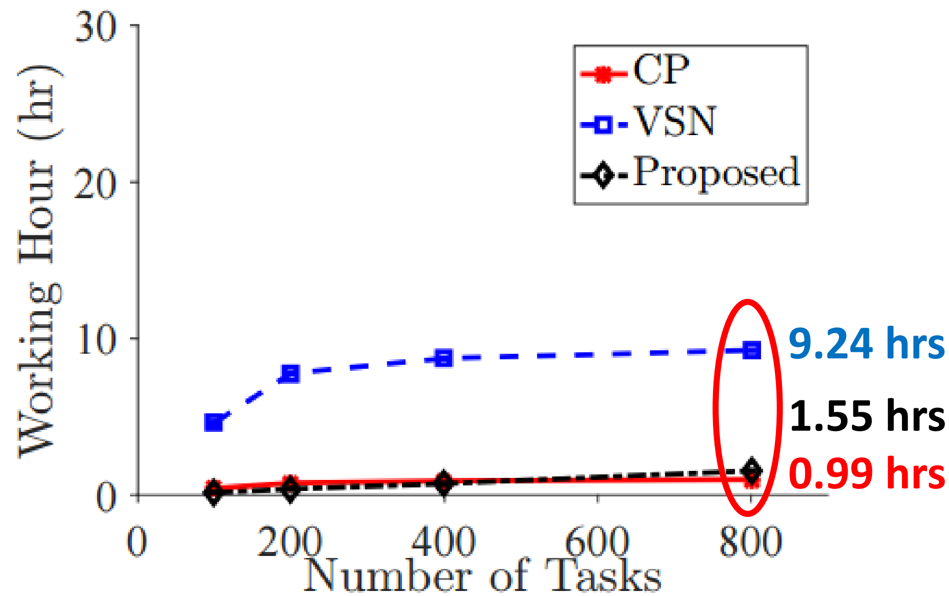
Working Hour

Random Way Point



Our algorithm reduces the average working hour by at most **87%**

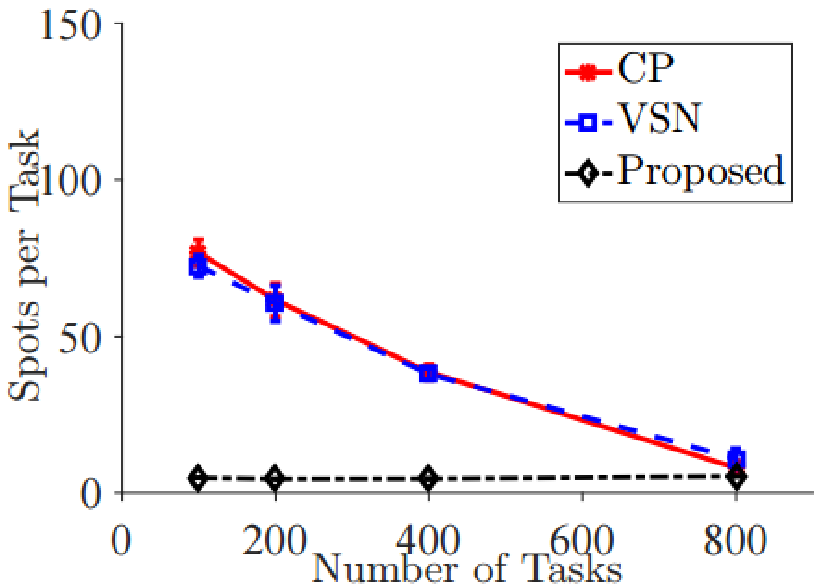
Pathway Mobility Model



The VSN algorithm have almost **9** times working hour compared to our solution

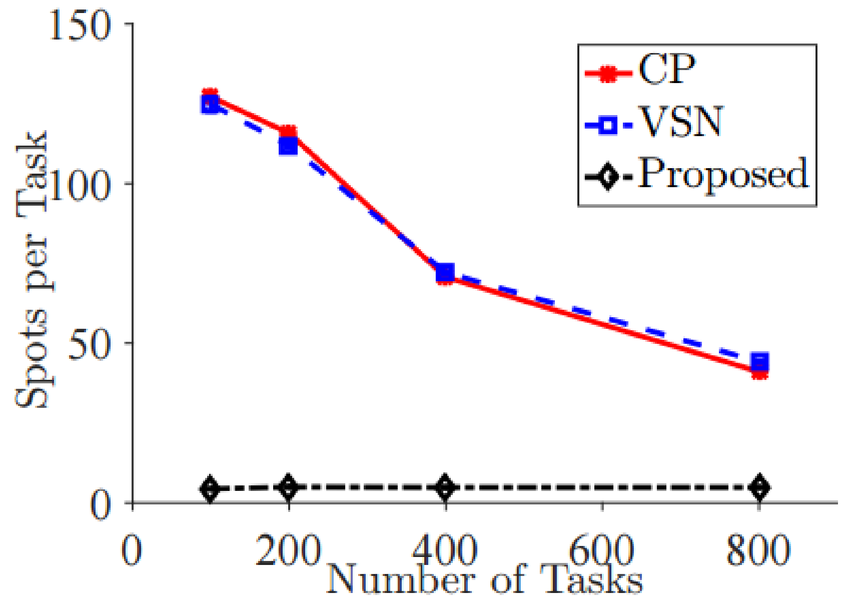
Spots Per Task

Random Way Point



Our algorithm produces stable spots per task

Pathway Mobility Model



The baseline algorithms end up with too many spots per task.

Running Time

- The running Time (μs) of Our Algorithms:
  - Spot Locator and Gamer Assigner are running on server.
  - NPC Path Generator is running on ASUS Zenfone 3

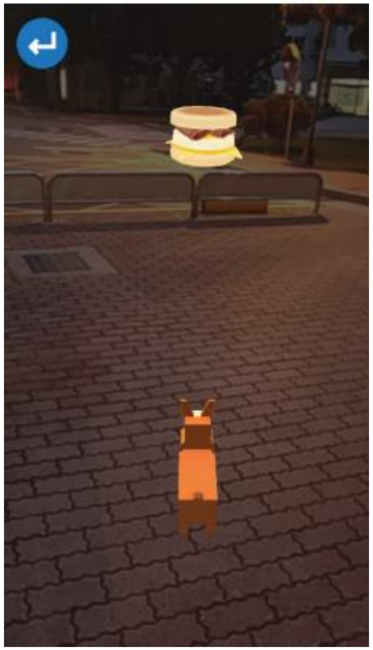
Algorithm	Running Time	
	Min	Max
Spot Locator	3161.85	3856.85
Gamer Assigner	55.12	57.93
NPC Path Generator	76.51	121.19

Outline

- Motivation
- Challenges & Approaches
  - Incentive Mechanism
  - Task Coverage
- Problems & Solutions
  - Optimal Spot Locator
  - Nearest Gamer Assigner
  - Nature NPC Path Generator
- Evaluations
  - Simulation Settings
  - Mobility Models
  - Baseline Algorithms
  - Simulation Results
- Implementation & User Study
- Conclusion

Implementation

- Two Android applications
  - Ordinary version
  - Gamified version



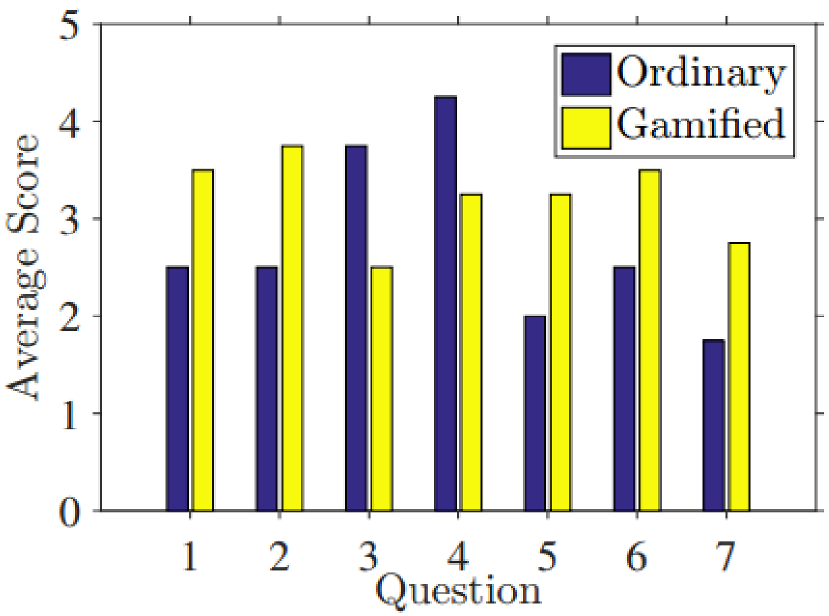
User Study

- The tasks are randomly generated every 5~30 minutes
- These tasks are located in 15 predefined locations in the campus
- We recruit 4 gamers in their twenties (50% male)
- We give them phones (Zenfone 3) for 3 days

User Study Results

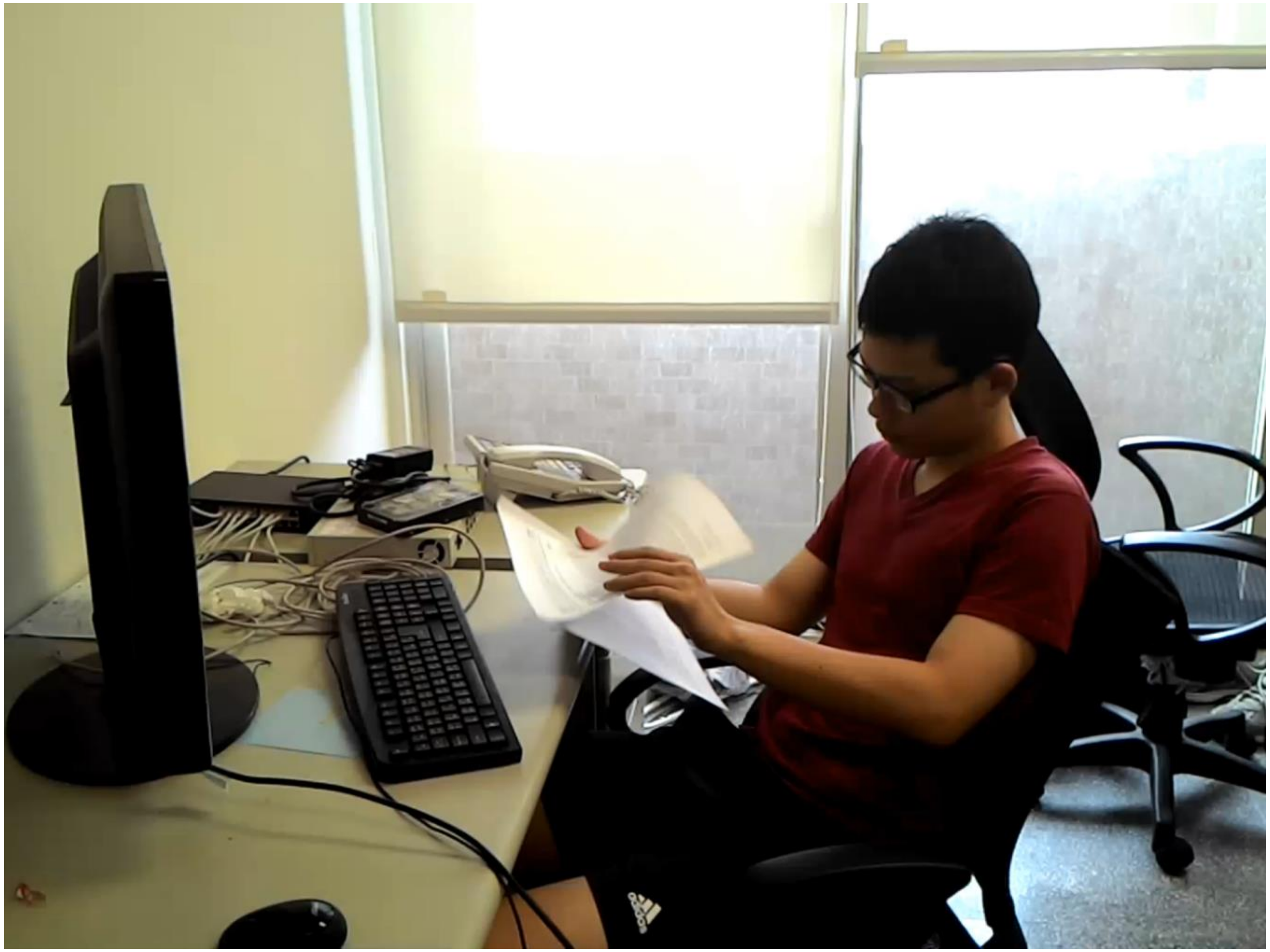
- We use five-point Likert scale (between 1 and 5) to assess the user study results.
- Intrinsic Motivation Inventory (IMI) [3]

	IMI questions
1	I enjoy this game
2	the game is fun
3	I think the game is boring
4	playing the game doesn't hold my attention
5	I would describe the game is interesting
6	the game is enjoyable
7	when I play the game, I think about how much I enjoy it



[3] E. Deci, H. Eghrari, B. Patrick, and D. Leone, "Facilitating internalization: The self-determination theory perspective," Journal of Personality, vol. 62, no. 1, pp. 119–142, 1994.

Demo



Outline

- Motivation
- Challenges & Approaches
    - Incentive Mechanism
    - Task Coverage
- Problems & Solutions
    - Optimal Spot Locator
    - Nearest Gamer Assigner
    - Nature NPC Path Generator
- Evaluations
    - Simulation Settings
    - Mobility Models
    - Baseline Algorithms
    - Simulation Results
- Implementation & User Study
- Conclusion

Conclusion

- Implement crowdsourcing system
  - Platform
  - Ordinary and gamified app
- We study the problem of gamifying mobile apps to transparently assign sensing tasks to gamers
  - Propose three algorithms
- Conduct the experiments using simulation and user study
  - The completion ratio of our algorithm outperforms the baseline algorithms by up to **74%**
  - Our algorithm produces stable spots per task
  - Gamification stimulates the gamers to complete tasks.

Thank You for Listening