A Detour Planning Algorithm in Crowdsourcing Systems for Multimedia Content Gathering

Chen-Chih Liao
Department of Computer Science
National Tsing Hua University
Hsin Chu, Taiwan

Cheng-Hsin Hsu
Department of Computer Science
National Tsing Hua University
Hsin Chu, Taiwan

ABSTRACT

Crowdsourcing is a popular paradigm that outsources tasks to general publics. In crowdsourcing systems, *requesters* submit requests and *workers* perform selective requests for rewards. We propose a crowdsourcing system for multimedia content gathering, which is a new class of crowdsourcing systems with requests that must be performed at specific locations and time. The workers in our systems gather multimedia content using their smartphones and share the collected content over the Internet. Our system computes a detour path for each worker, who supplies his/her destination with a deadline to the system, and do not mind to take detour paths and perform some requests for profits. More precisely, we develop a detour planning algorithm to produce the optimal detour paths for individual workers. Using extensive trace-driven simulations, we demonstrate the effectiveness and efficiency of our algorithm: for example, it helps workers to achieve optimal profits (up to $\sim 100\%$ improvement compared to baseline solutions) and runs in real-time (< 82 ms). Developing a working prototype on Android OS and addressing other challenging aspects of the considered systems are among our future tasks.

Categories and Subject Descriptors

H.5 [**Information Systems Applications**]: Multimedia Information Systems

General Terms

Algorithms

1. INTRODUCTION

Crowdsourcing refers to a distributed problem solving paradigm, in which some companies make *tasks*, traditionally performed by their employees, public to individuals who are willing to complete them for *rewards*. The companies are called *requesters*, and the individuals are called *workers*. Furthermore, each task is associated with a *deadline* and a *reward* amount. A worker is rewarded for a task only if he/she completes that task by its deadline. In crowdsourcing systems, workers are economically incentivized by the rewards of completing tasks.

In the past few years, a large body of crowdsourcing research has been done [22, 23]. The ever increasing popularity of crowdsourcing is largely because humans are good at many tasks that are challenging to computers and algorithms. For example, relevance evaluation, opinion collections, common sense derivations, and multimedia tagging are difficult to computers, but relatively easy for humans [23]. Moreover, leveraging on a huge number of humans, who may become workers, crowdsourcing often reduces the cost of completing the tasks, and thus leads to new business opportunities.

Because of the above-mentioned strengths, crowdsourcing has been employed in various applications, which can be grouped into four classes [23]: (i) voting systems, (ii) information sharing systems, (iii) creative systems, and (iv) social games. Voting systems, such as Amazon Mechanical Turk [1], dictate each worker to answer online questionnaires in order to find answers from the majority of the crowd. Information sharing systems allow the crowd to share knowledge, such as regional noise level [9] and network events [4]. Creative systems recruit workers to jointly build art creations. Social games entertain workers while generating some by-product results potentially useful for other applications. For example, geospatial tagging games with smartphones [3,7,11] require a worker to move to specific locations tagged by other workers in order to gain some points in the games. The main objective of these games is to entertain the workers and the games are driven by the workers themselves, rather than toward global benefits of the crowdsourcing systems.

We study a new class of crowdsourcing systems for serious applications of multimedia content gathering. In the considered systems, requesters submit geospatial- and temporal-dependent tasks to collect multimedia content, such as sensor readings from sparely-deployed nodes, recorded videos of specific events, and photos of sightseeing sites. The corresponding requesters could be utility companies who need to retrieve smart-meter readings, police departments who need to collect evidence of crime scenes, and people who need photos of memorable locations for pleasure, respectively. The workers are smartphone users who are heading to their final *destinations*, but have some time to spare. The workers wouldn't mind to take some *detour paths*, for small rewards, which can be monetary, credits, or points, as long as they can reach the destinations in time.

The considered systems are unique because tasks are associated with deadlines and geospatial locations, while it takes workers time to travel from the location of one task to that of another one. Moreover, existing crowdsourcing systems require workers to *compete* against one another for tasks. We argue that the same approach

MoVid'13, February 26-March 1, 2013, Oslo, Norway.

is not effective for our crowdsourcing systems because: (i) while workers have time to take detour paths, they are not capable to come up with the best detour paths and (ii) too many competitions may lead to excessive duplicated efforts and degraded overall system performance.

In this paper, we design a *detour planning algorithm* for our crowdsourcing systems. The proposed algorithm runs on a server, and generates a detour path for each worker. The resulting detour path is optimal in the sense that it maximizes his/her profit, while guaranteeing the worker can reach his/her destination in time. We implement the proposed algorithm in a simulator, and compare its performance against several baseline algorithms. We also crawl the location traces of a large number of Flickr [5] photos, and use the traces to drive the simulator. The simulation results show that our algorithm: (i) outperforms the other algorithms by up to 100% improvement, (ii) runs efficiently and always terminates in 82 ms, and (iii) scales to large problems.

The rest of this paper is organized as follows. Sec. 2 surveys the literature. We define the considered problem in Sec. 3. The optimal algorithm is proposed in Sec. 4. This is followed by the trace-driven simulations in Sec. 5. Sec. 6 discusses the future directions, and Sec. 7 concludes the paper.

2. RELATED WORK

The crowdsourcing paradigm has been employed in the literature to solve various real-life problems. Many of these studies leverage smartphones for their high penetration rate. For example, Li et al. [8] design a distributed question and answer system, called SOS, for smartphone users. SOS employs crowdsourcing to find the answers for the questions that can not be addressed by search engines. It also leverages social networks to propagate the unanswered questions via friends' connections. Yan et al. [21] adopt crowdsourcing to develop an information sharing system, called CrowdPark. A driver who will need a parking spot soon may use his/her smartphone to reserve it in advance. For a driver who is vacating a parking spot, he/she can notify the CrowdPark server for selling the parking spot to incoming drivers. Crowdsourcing paradigm has also appeared in commercial systems. For example, Roamler [14] allows companies to define different tasks for iOS users. The users install an iOS application to receive and complete tasks so as to earn money or points. The Roamler server pushes tasks to users based on their preferences and locations. Different from our work, the works in [8, 21–23] and commercial systems [14] concentrate on building the crowdsourcing platforms, which facilitate interactions between the requesters and workers, but do not *guide* the workers for more efficient decisions.

Some other crowdsourcing systems guide the workers for better decisions. For example, Yuen et al. [24] study the task matching problem in crowdsourcing. Their system generates a task list for each worker based on his/her preference and historical performance, which helps workers to concentrate on completing tasks rather than searching for ideal tasks. Different from our work, the problem in [24] is not geospatial-dependent. Bassem and Bestavros [2] tackle a Geo-temporal Request Satisfaction (GRS) problem from a game-theoretic perspective, and propose a heuristic algorithm that works as follows. The algorithm derives a path from the starting location to the final destination using Yen's ranking algorithm [10], and chooses the task with the highest reward on this path. The set up in [2] is quite different from ours, because they allow workers to compete for tasks. In contrast, we believe that *coordinated* detour planning for minimizing wasted worker efforts is critical to the overall system performance. Moreover, we take the properties of mobile multimedia into considerations while designing the system. Last, their heuristic algorithm does not give optimal paths.

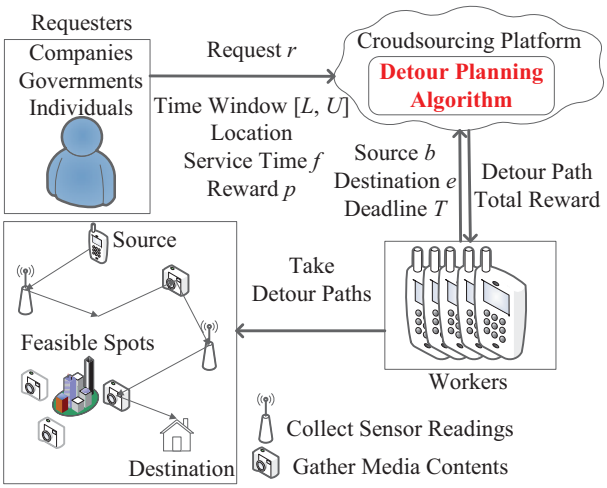

Figure 1: The considered detour planning problem. This paper concentrates on multimedia content gathering.

3. DETOUR PLANNING PROBLEM

Fig. 1 illustrates the considered crowdsourcing system, which consists of three entities: (i) the platform, (ii) requesters, and (iii) workers. Let N be the number of total requests that haven't been assigned to any workers. A requester sends request r_i ($1 \leq i \leq N$) with a time window $[L_i, U_i]$, reward p_i, location, and service time f_i to the platform using a web site or via a smartphone application. The requesters can be companies, governments, and individuals. If r_i is completed by a worker between time L_i and U_i, that worker receives reward p_i. A worker sends his/her current, or *source*, location b, destination location e, and deadline T to the crowdsourcing platform. Whenever there is a new worker, the crowdsourcing platform computes the detour path $\mathbf{X}$ for the worker using the *detour planning* (DP) algorithm proposed in Sec. 4. Moreover, a new detour path is generated if a worker is late or lost when following the previously computed detour path. The resulting detour path allows the worker to maximize his/her total reward, and ensures the worker to arrive the destination by the specific deadline. Upon receiving the detour path, a worker follows it to complete the individual requests.

Requesters may submit several types of requests. Fig. 1 shows two representative types: (i) sensor data collection and (ii) media data gathering. Examples of the latter type include taking a photo of a landmark, which may be done at multiple close-by locations referred to as *feasible spots*. We let $z_i \geq 1$ ($1 \leq i \leq N$) be the number of feasible spots of request r_i. We let f_i be the *service time* of request r_i, i.e., the amount of time a worker has to spend at r_i. To plan the detour paths, we also need to know that travel time between any two feasible spots. We let m_{i_x,j_y} be the travel time from feasible spot x of request r_i to feasible spot y of request r_j, and collectively write this distance matrix as $\mathbf{M}$. Similarly, c_{i_x,j_y} and $\mathbf{C}$ represent the *cost* of traveling from a request to another, which can be attributed to gas cost and car depreciation rate. Both $\mathbf{M}$ and $\mathbf{C}$ are predetermined, for example, by using Google Map and other online maps.

Next, we define the decision variables and some intermediate variables for the considered detour planning problem. We use boolean variables to represent the path $\mathbf{X}$. Specifically, $x_{i,j} = 1$ if

the detour path leads a worker moving from request i to j; and $x_{i,j} = 0$ otherwise. We also use s_i to represent the worker's planned arrival time at the location of request r_i. We let $u_{i,j} = 1$ ($1 \leq i \leq n$, $1 \leq j \leq z_i$) if feasible spot j of request r_i is on the detour path; and $u_{i,j} = 0$ otherwise.

Inspired by the formulations given in Vansteenwegen et al. [20], we mathematically formulate our detour planning problem as:

$$\max \sum_{i=1}^{N-1} \sum_{j=2}^{N} [p_i - c_{ij}()] x_{i,j} \quad (1)$$

$$s.t. \sum_{j=2}^{N} x_{1,j} = \sum_{i=1}^{N-1} x_{i,N} = 1 \quad (2)$$

$$\sum_{i=1}^{N-1} x_{i,k} = \sum_{j=2}^{N} x_{k,j} \leq 1, \forall k = 2, ..., N-1 \quad (3)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} m_{i_x,j_y} x_{i,j} \leq T_{max} \quad (4)$$

$$\sum_{j=1}^{z_k} u_{k,j} \leq 1, \forall k = 1, ..., N \quad (5)$$

$$L_i \leq s_i, \forall i = 1, ..., N \quad (6)$$

$$s_i + f_i \leq U_i, \forall i = 1, ..., N \quad (7)$$

$$x_{i,j}, u_{i,j} \in \{0, 1\}. \quad (8)$$

The objective function Eq. (1) is to maximize the total collected reward minus travel cost, which is referred to as profit. The constraints in Eq. (2) ensure that the path starts from request 1 to request N. The constraints in Eq. (3) make sure that every feasible spot is visited once. The constraints in Eq. (4) ensure that the total time of each path doesn't exceed the deadline specified by each worker. The constraints in Eq. (5) make sure that only one feasible spot of each request is visited. The constraints in Eqs. (6) and (7) consider the worker's arrival time and service time.

Hardness of our problem. The detour planning problem is a *generalized* version of the *orienteering problem* (OP), which computes a path to visit some locations in order to maximize the profit and arrive at the destination in time. Golden et al. [6] show that OP problems are NP-hard, and Vansteenwegen et al. [20] present four OP variations: (i) single worker, (ii) multiple workers, (iii) single worker with time-windowed requests, and (iv) multiple workers with time-windowed requests. The first two variations (without time-windowed requests) have been well studied in the literature, e.g., Schilde et al. [15] and Vansteenwegen et al. [19] propose algorithms to solve single- and multiple-worker OP problems without time-windowed requests. To our best knowledge, OP problems with time-windowed requests have not been throughly studied. The two most recent works are: Righini and Salani [13] and Montemanni and Gambardella [12], which solve the single- and multiple-worker variations, respectively.

The considered detour planning problem (Eqs. (1)–(8)) is close to the OP problem with single worker and time-windowed requests. However, our crowdsourcing problem has the following unique features:

1. Each request r_i is associated with a non-trivial service time f_i. For example, it may take a worker 3 minutes to download the sensor readings or take a photo.
2. Each request r_i may be satisfied from z_i feasible spots. $z_i = 1$ if the request can only be performed at a specific location.
3. Relocating a worker from request r_i to r_j imposes nontrivial cost $c_{i,j}$ on that worker.

Therefore, the existing OP solutions cannot be applied to our problem. We develop a detour planning algorithm in Sec. 4.

Detour Planning (DP) Algorithm

1: **Let** $\alpha_{i,j} = < \mathbf{V}_\emptyset, 0, 0, \mathbf{u}_\emptyset >, \forall \alpha_{i,j} \in \chi$
2: **Let** $\alpha_{1,1}.\mathbf{V} = \alpha_{1,1}.\mathbf{V} \cup \{v_{1,1}\}$; $\alpha_{1,1}.\mathbf{u} = \alpha_{1,1}.\mathbf{u} \cup \{u_{1,1} = 1\}$
3: **Enqueue** $\alpha_{1,1}$ into $\mathbf{E}$ //E: vertices to visit
4: **while** $\mathbf{E} \neq \emptyset$ **do**
5: **Dequeue** $\alpha_{i,x} \in \mathbf{E}$
6: //Extend the $\alpha_{i,x}$ to a new $\alpha_{j,y}$
7: **for** each neighboring $\alpha_{j,y} \in \chi \backslash \{\alpha_{i,k}\}_{1 \leq k \leq z_i}$ **do**
8: **Let** $temp.\mathbf{V} = \alpha_{i,x}.\mathbf{V} \cup \{v_{j,y}\}$; $temp.w = \alpha_{i,x}.w + p_j - c_{i_x,j_y}()$; $temp.e = \alpha_{i,x}.e + m_{i_x,j_y} + f_j$; $temp.\mathbf{u} = \alpha_{i,x}.\mathbf{u} \cup \{u_{j,y} = 1\}$ //extand the path
9: **if** $temp.e \leq U_j$ **then**
10: **if** $temp.w \geq \alpha_{j,y}.w$ **then**
11: **Let** $\alpha_{j,y} = temp$; **Enqueue** $\alpha_{j,y}$ into $\mathbf{E}$
12: **Let** $\mathbf{X}^* = \alpha_{n,1}$

Figure 2: Pseudocode of finding the optimal path $\mathbf{X}^*$.

4. THE PROPOSED ALGORITHM: DP

We develop an optimal algorithm for our detour planning problem. Our Detour Planning (DP) algorithm is inspired by the dynamic programming algorithm presented in Righini and Salani [13], but we explicitly take the three unique features (see Sec. 3) into considerations. We let $\alpha_{i,j}$ be the state of a potential detour path from source location s to feasible spot j of request r_i. More specifically, we define $\alpha_{i,j} = < \mathbf{V}, w, e, \mathbf{u} >$, where $\mathbf{V}$ is the current path, w is the profit, e is the elapse time, and $\mathbf{u}$ keeps track of the visited requests and feasible spots. We let $\chi = \{\alpha_{i,j}\}_{1 \leq i \leq N, 1 \leq j \leq z_i}$ be the set of all the states with the maximum objective function values known so far. Fig. 2 presents the pseudocode of our algorithm, which consists of two steps: (i) initialization and (ii) expansion. In particular, lines 1–2 initialize all $\alpha_{i,j}$, and lines 4–11 iteratively extend $\alpha_{i,x}$ to $\alpha_{j,y}$. The optimal detour path is stored in $\alpha_{n,1}$, which is returned as $\mathbf{X}^*$ in line 12.

We analyze the efficiency of the proposed DP algorithm below.

REMARK 1. *The DP algorithm given in Fig. 2 has a time complexity of $O(N^3 Z^3)$, where Z is the maximum feasible spots for each request. That is $Z = \max_{1 \leq i \leq N} \{z_i\}$. This is because the dequeue command at line 5 may repeat for up to $N^2 Z^2$ times, as each edge may only be added to queue $\mathbf{E}$ once. The for-loop starts from line 7 has a complexity of $O(NZ)$ as it may check all the states. Hence, the time complexity of the DP algorithm is $O(N^3 Z^3)$.*

We next briefly explain a possible optimization that can be applied to the DP algorithm.

REMARK 2. *The proposed DP algorithm can be optimized in various ways. For example, Righini and Salani [13] propose a bidirectional approach, in which they use DP algorithm from both sides, the beginning and the end. Each direction extends the path and stops right before exceeding the half of the total number of requests. Then, the two detour paths are merged into an optimal detour path.*

5. EVALUATION

In this section, we conduct trace-driven simulations to evaluate the proposed DP algorithm.

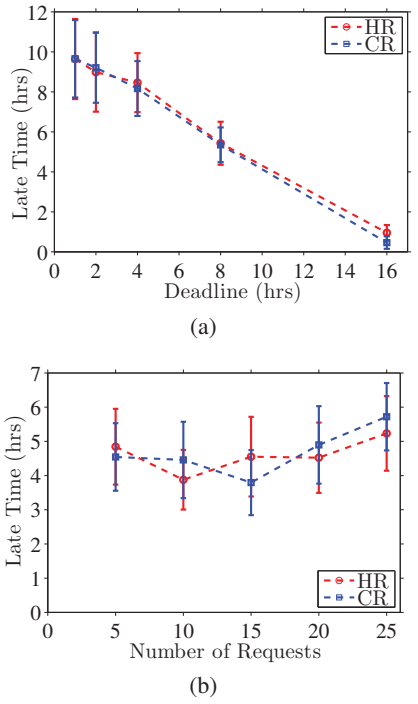

Figure 3: Late time of the HR and CR algorithms: (a) varying deadlines in Taipei and (b) varying numbers of requests in Vancouver.

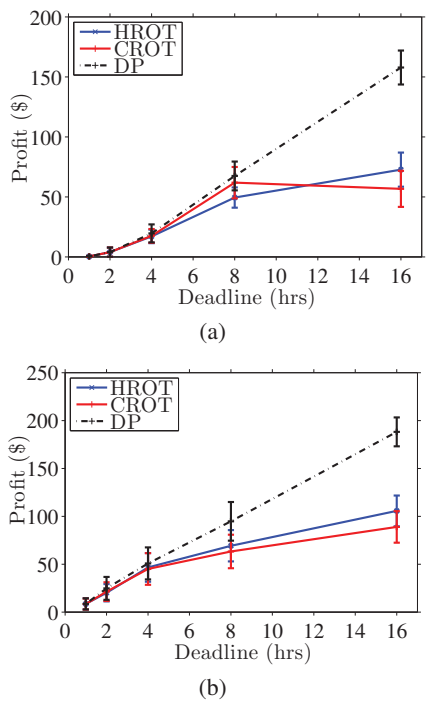

Figure 4: Resulting profit with varying deadlines, from: (a) Taipei and (b) Vancouver.

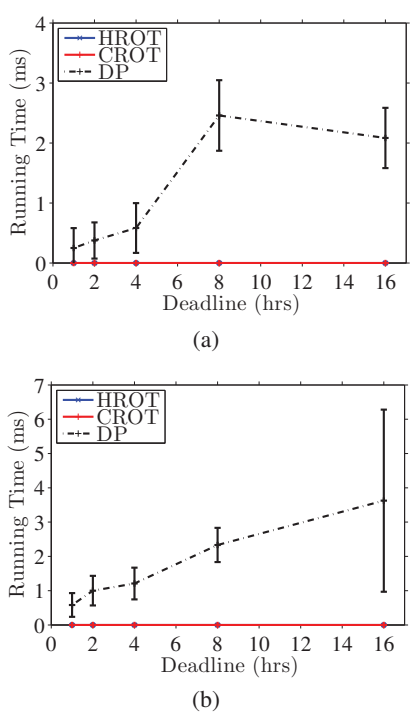

Figure 5: Running time with varying deadlines, from: (a) Taipei and (b) Vancouver.

Table 1: Ontime Ratio (%) of Various Algorithms

City	Taipei			Vancouver		
Algorithm	**HR**	**CR**	**DP**	**HR**	**CR**	**DP**
Deadline $T = 1$	0	0	100	0	0	100
2	4.1	4.1	100	4.1	0	100
4	0	0	100	0	0	100
8	0	0	100	0	4.1	100
16	29.1	58.3	100	33.3	41.6	100
City	**Taipei**			**Vancouver**		
Algorithm	**HR**	**CR**	**DP**	**HR**	**CR**	**DP**
No. Requests $N = 5$	12.5	8.3	100	0	0	100
10	0	0	100	0	4.1	100
15	0	8.3	100	0	0	100
20	0	0	100	0	0	100
25	0	4.1	100	0	0	100

5.1 Setup

We have developed a simulator for the detour planning problem using C/C++, and run the simulations on a commodity PC with an AMD 2.6 GHz CPU. Within the simulator, we have implemented the proposed DP algorithm[1]. We are not aware of any algorithms solving the considered problem, thus we have also implemented four heuristic algorithms, Highest-Reward (HR), Highest-Reward with OnTime constraints (HROT), Closest-Request (CR) and Closest-Request with OnTime constraints (CROT), for comparisons. The HR algorithm works as follows. It gives requests with higher rewards higher priority, and iteratively extends the detour path to the request with the highest priority. It stops when adding the next request renders the worker missing his/her deadline. The CR algorithm works in a similar way, but gives the closest request higher priority. The HR and CR algorithms mimic human behaviors when no algorithm is used to generate detour paths. We then create the ontime versions (HROT and CROT) of the algorithms, which validate the ontime constraint before extending a detour path to the next request. In particular, HROT and CROT do not consider request r_i unless the worker may travel from r_i to e before time T. Hence, HROT and CROT guarantee that workers can reach their destinations in time.

To drive our simulator, we consider the actual requests of multimedia content gathering, and we collect actual geospatial traces from Flickr [5] as follows. We first collect the names of 25 attractions from travel Web sites. In particular, Taipei [17] and Vancouver [18] are considered in our evaluations. We then look up the longitude/latitude of each attraction, and search for Flickr photos that are taken at close-by longitude/latitude and are tagged with the attraction's name. By close-by, we refer to the photos taken within 1 km radius of each attraction. We end up with having up to 2000 photos for each attraction. We extract the longitudes/latitudes from individual photos, and cluster them into a few feasible spots using hierarchical clustering with a threshold of 500 meters. The mean longitude/latitude of each cluster is used as the location of a feasible spot. The number of feasible spots for each attraction is between 1 and 31, depending on the attraction's height and popularity.

Each simulation lasts for 25 hours. We assume the requests follow a Poisson arrival process and we set the average number

[1]We thank the authors of [13] for sharing their datasets and algorithm implementation with us.

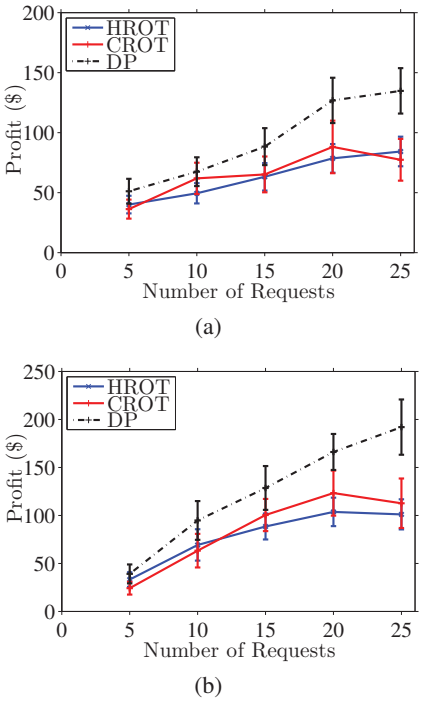

Figure 6: Resulting profits with varying numbers of requests, from: (a) Taipei and (b) Vancouver.

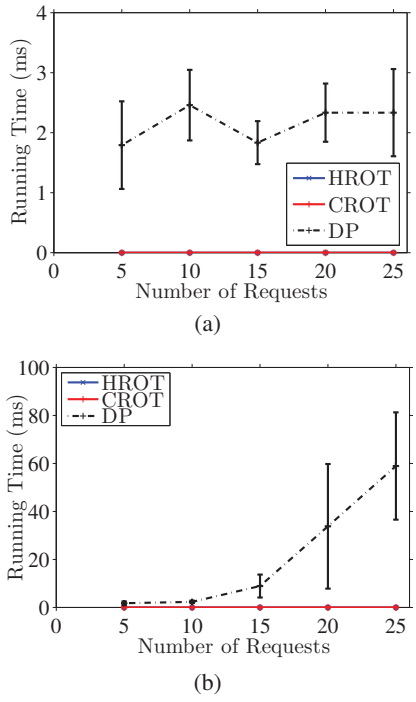

Figure 7: Running time with varying numbers of requests, from: (a) Taipei and (b) Vancouver.

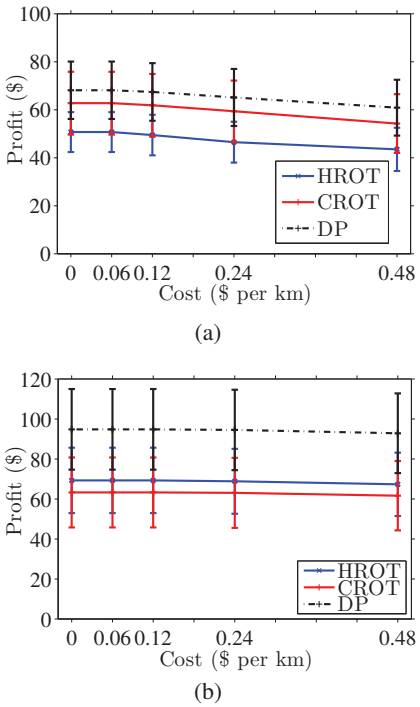

Figure 8: Resulting profits with different travel cost, from: (a) Taipei and (b) Vancouver.

of requests per hour to be 4. Each request happens at a random attraction, with a time window size between 1–6 hours and a reward between 1–40 U.S. dollars, both follow uniform distributions. We vary three system parameters in the simulations. $T \in \{1, 2, 4, 8, 16\}$ is the deadline of arriving at the destination, $N \in \{5, 10, 15, 20, 25\}$ is the number of attractions, and $C \in \{0, 0.06, 0.12, 0.24, 0.48\}$ dollars per km is the average gas and car depreciation cost. We let $T = 8$, $N = 10$, and $C = 0.12$ if not otherwise specified. We run each simulation 24 times, and report the simulation results with 95% confidence intervals whenever applicable. We consider four performance metrics: (i) *late time* of HR and CR, (ii) *ontime ratio*, which is the fraction of workers who reach their destinations in time, (iii) *profit* of the resulting detour path, and (iv) *running time* of each algorithm.

5.2 Results

Importance of the DP problem. In Fig. 3, We first report sample late time from the HR and CR algorithms, which mimic human behavior. This figure reveals that the HR and CR algorithms lead to up to almost 10 hrs of average late time. This is significant considering that the deadline of arriving at the destinations is as short as 1 hr. Long late time will drive the potential workers away from the crowdsourcing systems. We next present the ontime ratio of HR, CR, and DP algorithms in Table 1. This table clearly shows the benefit of the DP algorithm: the computed detour paths are always ontime. Fig. 3 and Table 1 depict the importance of the considered DP problem, as human computed detour paths (mimicked by HR and CR) lead to late arrivals at the destinations. Since the HR and CR algorithms do not meet the basic requirement of the DP problem, we no longer consider them in the rest of this paper.

Optimized profits. Fig. 4 presents the profits of different algorithms under varying deadlines. This figure reveals that the DP algorithm always outperforms the HROT and CROT algorithms. Moreover, the performance gain of the DP algorithm over the other two algorithms increases with longer deadlines. In particular, with $T = 16$, the DP algorithm almost doubles the profits, compared to the other two algorithms.

Efficiency. Fig. 5 plots the running time of the considered algorithms. We observe that the heuristic algorithms finish in no time, and the DP algorithm takes about 6 ms to terminate in the worst case, which shows its efficiency.

Scalability of the DP algorithm. We present the simulation results with diverse numbers of requests in Figs. 6 and 7, for resulting profits and running time, respectively. These two figures demonstrate that the DP algorithm scales to large numbers of requests and feasible spots quite well. In Taipei, the average number of feasible spots per-request is 1.96 and the maximal number of feasible spots per-request is 15. In Vancouver, the average and maximal numbers of feasible spots per-request are 6.48 and 31, respectively. Fig. 6 reveals that the DP algorithm outperforms all other algorithms under all considered numbers of requests. Fig. 7 depicts that the DP algorithm terminates much slower in Vancouver (Fig. 7(b)) than in Taipei (Fig. 7(a)): up to 82 ms running time is observed in Vancouver. This difference can be explained by the fact that the attractions in Vancouver have more feasible spots (162 in total), compared to the attractions in Taipei (49 in total), as reported above. We conclude that the DP algorithm does scale to 162 feasible spots, yet runs in < 82 ms, which essentially is in real-time.

Implication of cost. Fig. 8 reports the sample resulting profits from the algorithms with different travel cost. This figure reveals that, when the per-km cost is lower, the proposed DP algorithm

results in higher profits. Moreover, the DP algorithm outperforms the heuristic algorithms under all considered travel costs.

6. CONTRIBUTIONS AND FUTURE DIRECTIONS

We are actively developing a complete crowdsourcing system for multimedia content gathering. The following challenges arise when designing such a geospatial- and temporal-dependent system.

1. The systems should produce a *feasible* detour path for each new worker. A detour path is feasible if and only if the worker can reach his/her destination in time.
2. The systems should compute the detour paths to maximize the overall productivity in the format of the total worker profit.
3. The systems should guide the workers to shoot the photos or videos from the right spots and angles for usable results. The locations can be determined by various proposals in the literature, e.g., Shen et al. [16] develop a system to predict the viewable scenes of any given longitude and latitude.
4. The systems should adapt to misbehaved or lost workers who deviate from the assigned detour paths, by producing new detour paths for them.
5. The systems should facilitate a quality assurance mechanism to determine the quality of the completed tasks and to detect cheating.

We address the first two challenges in this paper. The remaining three challenges, as well as other practical concerns, are our future tasks.

7. CONCLUSIONS

In this paper, we studied the detour planning problem in a crowdsourcing system employing geospatial- and time-dependent requests with time windows and service time. In the considered system, workers pay for gas and car depreciation to travel among requests' locations. We proposed the DP algorithm, which generates the optimal detour path for each worker. The detour path is optimal in the sense that the worker makes the highest profit without being late at the destination. We conducted through trace-driven simulations to evaluate the proposed DP algorithm. The simulation results reveal that, the DP algorithm: (i) guarantees ontime arrivals at the destinations, (ii) increases the profit by up to about 100%, (iii) runs fast and terminates in 82 ms, and (iv) scales to large problems with up to 162 feasible spots. We are currently implementing the proposed crowdsourcing system on real smartphones and servers. The resulting prototype system can be used by police departments for collecting evidence, individuals for gathering sight-seeing photos, and utility companies for retrieving smart-meter reading. Various challenges in the prototype system will be studied in our future work.

8. REFERENCES

[1] Amazon mechanical turk, July 2012. `https://www.mturk.com`.

[2] C. Bassem and A. Bestavros. Mechanism design for spatio-temporal request satisfaction in mobile networks. Technical report, Boston University, February 2012.

[3] S. Casey, B. Kirman, and D. Rowland. The gopher game: A social, mobile, locative game with user generated content and peer review. In *Proc. of ACM International Conference on Advances in Computer Entertainment Technology (ACE'07)*, pages 9–16, Salzburg, Austria, June 2007.

[4] D. Choffnes, F. Bustamante, and Z. Ge. Crowdsourcing service-level network event monitoring. In *Proc. of ACM SIGCOMM'10*, pages 387–398, New Delhi, India, August 2010.

[5] flickr, October 2012. `http://www.flickr.com/`.

[6] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307–318, 1987.

[7] L. Grant, H. Daanen, S. Benford, A. Hampshire, A. Drozd, and C. Greenhalgh. MobiMissions: The game of missions for mobile phones. In *Proc. of ACM SIGGRAPH'07*, San Diego, CA, August 2007.

[8] Z. Li, H. Shen, G. Liu, and J. Li. SOS: A distributed mobile Q&A system based on social networks. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS'12)*, Macau, China, June 2012.

[9] N. Maisonneuve, M. Stevens, M. Niessen, P. Hanappe, and L. Steels. Citizen noise pollution monitoring. In *Proc. of International Digital Government Research Conference (DG.O'09)*, pages 96–103, Puebla, Mexico, May 2009.

[10] E. Martins and M. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR: A Quarterly Journal of Operations Research*, 1(2):121–133, June 2003.

[11] S. Matyas, C. Matyas, C. Schlieder, P. Kiefer, H. Mitarai, and M. Kamata. Designing location-based mobile games with a purpose: Collecting geospatial data with CityExplorer. In *Proc. of ACM International Conference on Advances in Computer Entertainment Technology (ACE'08)*, pages 244–247, Yokohama, Japan, December 2008.

[12] R. Montemanni., D. Weyland, and L. Gambardella. An enhanced ant colony system for the team orienteering problem with time windows. In *Proc. of International Symposium on Science and Society (ISCCS'11)*, pages 381–384, Kota Kinabalu, Malaysia, July 2011.

[13] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers and Operations Research*, 36(4):1191–1203, April 2009.

[14] Roamler, 2011. http://www.roamler.com/services.aspx.

[15] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence*, 3(3):179–201, September 2009.

[16] Z. Shen, S. Ay, S. Kim, and R. Zimmermann. Automatic tag generation and ranking for sensor-rich outdoor videos. In *Proc. of ACM International Conference on Multimedia (MM'11)*, pages 93–102, Scottsdale, AZ, November 2011.

[17] Taipei travel net, October 2012. `http://www.taipeitravel.net/en/`.

[18] Vancouver landmarks, October 2012. `http://www.hotels.com/de169712-la/all-landmarks-in-vancouver-canada/`.

[19] P. Vansteenwegena, W. Souffriaua, G. Berghe, and D. Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118–127, July 2009.

[20] P. Vansteenwegena, W. Souffriaua, and D. Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, February 2011.

[21] T. Yan, B. Hoh, D. Ganesan, K. Tracton, T. Iwuchukwu, and J. Lee. CrowdPark: A crowdsourcing-based parking reservation system for mobile phones. Technical report, University of Massachusetts Amherst, 2011.

[22] M. Yuen, L. Chen, and I. King. A survey of human computations systems. In *Proc. of IEEE Symposium on Social Computing Applications (SCA'09)*, pages 723–728, Vancouver, Canada, October 2009.

[23] M. Yuen, I. King, and K. Leung. A survey of crowdsourcing systems. In *Proc. of IEEE International Conference on Social Computing (SocialCom'11)*, pages 766–773, Boston, MA, October 2011.

[24] M. Yuen, I. King, and K. Leung. Task matching in crowdsourcing. In *Proc. of IEEE International Conference on Internet of Things, and Cyber, Physical and Social Computing (iThings/CPSCom'11)*, pages 409–412, Dalian, China, October 2011.