Mechanism Design for Spatio-Temporal Request Satisfaction in Mobile Networks

Christine Bassem and Azer Bestavros

February 10, 2012

Abstract

Mobile agents participating in geo-presence-capable crowdsourcing applications should be presumed rational, competitive, and willing to deviate from their routes if given the right incentive. In this paper, we design a mechanism that takes into consideration this rationality for request satisfaction in such applications. We propose the Geo-temporal Request Satisfaction (GRS) problem to be that of finding the optimal assignment of requests with specific spatio-temporal characteristics to competitive mobile agents subject to spatio-temporal constraints. The objective of the GRS problem is to maximize the total profit of the system subject to our rationality assumptions. We define the problem formally, prove that it is NP-Complete, and present a practical solution mechanism, which we prove to be convergent, and which we evaluate experimentally.

1 Introduction

We refer to the capability of an application to access devices at particular geographical locations and particular times as *Geo-Presence*. Typically, geo-presence is supported through the use of dedicated agents operating in proprietary cyber-physical infrastructures, such as embedded sensors and actuators, or input and output devices that are mounted in physical spaces. Clients of geo-presence-capable systems submit spatio-temporal requests, which are assigned by the system to mobile agents who are able to service such requests. We categorize geo-presence-capable systems based on the mobility characteristics of the agents as either *infrastructure-based* or *crowdsourcing-based* systems. In infrastructure-based geo-presence-capable systems, agents are either stationary such as in wireless sensor networks [23, 3], or have limited system-controlled mobility such as in field coverage using robotics [11, 42, 45]. Alternatively, in crowdsourcing-based geo-presence-capable systems [4, 1], agents must be presumed autonomous and rational, in the sense that they control their own mobility and schedules.

Our work considers the second class of geo-presence-capable systems – namely, crowdsourcing-based systems in which requests submitted by clients are outsourced to mobile users who subscribe as willing agents of the system through

an open interface. Depending on the application, requests can range from taking pictures of certain locations and at particular times to displaying advertisements in certain locations at particular times. Moreover, we assume the subscribed agents to be rational, and willing to deviate from their routes if given the right incentive. Based on these assumptions, we consider a *Geo-Presence as a Service* (GPaaS) framework, which provides market-place crowdsourcing services using the help of already existing mobile agents in the field. The framework acts as a proxy between clients with spatio-temporal requests and agents capable of satisfying these requests. The service provided by the GPaaS framework is on-demand and is non-archival, *i.e.*, data is only collected when requested and is not stored for future use. The services provided are market-place priced according to the supply and demand in the system, and are pay-as-you-go, *i.e.*, clients only pay for services they requested over a given duration.

Although the request satisfaction problem in such a system may seem as a traditional scheduling problem, the spatio-temporal constraints of the agents and their willingness to deviate from their paths introduce a new class of scheduling problems, in which the agents' mobility can be managed. In this paper, we formally define the Geo-temporal Request Satisfaction (GRS) problem as that of finding the optimal assignment of requests with specific spatio-temporal characteristics to competitive mobile agents subject to spatio-temporal constraints. The objective of the GRS problem is to maximize the total profit of the system. We define the problem formally, prove that it is NP-Complete, and present a practical solution mechanism, which we prove to be convergent, and which we evaluate experimentally.

Paper Outline. In Section 2, we define the Geo-temporal Request Satisfaction problem and prove its NP-Completeness. In Section 3, we propose a heuristic game-theoretic approach for solving the problem and analyze it analytically. In Section 4, we evaluate its performance experimentally using simulations. In Sections 5 and 6, we conclude with a discussion of the related work and of our current and future work.

2 Geo-temporal Request Satisfaction Problem

2.1 Problem Definition

We model the structure of the mobility field (e.g., map of city or locale) as a graph $G = (V, E)$ in which the set of vertices V represents the various landmarks in the field (e.g., intersections) and the set of edges E represents the links between these landmarks (e.g., streets). We denote by R the set of requests submitted to the system, and by A the set of agents participating in the system.

A request in R is defined by the 3-tuple $(v, t, p())$ where $v \in V$ is the desired location of the request, $t \geq 0$ is its desired time, and $p(v', t')$ is its valuation function. The valuation of the request is maximum at the desired location v and time t, and may differ otherwise. In particular, for $v' \in V$ and $t' \geq 0$, the valuation of the request is given by $p(v', t')$.

An agent in A is defined by the 5-tuple $(v_o, t_o, v_f, t_f, c())$ representing the *journey* of the agent and its cost function. The agent's desired journey is defined by its starting location $v_o \in V$ and time $t_o \geq 0$, as well as the desirable target location $v_f \in V$ and latest arrival time to that location $t_f > t_o$. The agent's cost function $c(p_j)$ defines the cost incurred by the agent when choosing the path p_j to make its desired journey.[1]

Definition 1 *(The GRS Problem) Given the mobility field graph G, a list of requests R, and a list of agents A, the GRS problem is that of finding a legitimate path for each agent in the list A that maximizes the total profit of the system. The total profit of the system is defined as the difference between the total valuation obtained from the serviced requests as defined by their valuation functions and the total cost incurred by the agents servicing these requests as defined by their cost functions. Moreover, a legitimate path has to satisfy the journey constraints defined by the agent,* i.e.*, start at its desired start location and time and end at its destination at time $t < t_f$.*

2.2 GRS problem is NP-Complete

We prove that the GRS problem is NP-Complete by reducing it to the Hamiltonian Cycle problem. A Hamiltonian cycle is a cycle in an undirected graph that visits each vertex exactly once and returns to the starting vertex. Given any graph, determining whether a Hamiltonian cycle exists is known to be NP-complete [15].

Theorem 2 *The GPS problem is NP-Complete.*

Proof. Consider an instance of the GRS problem where,

- The mobility field graph is defined as $G = (V, E)$ with $|V| = l$.
- The list of requests R has l requests, in which each request r_i is defined by the tuple $(v_i, 0, p(v', t'))$ and,

$$p(v', t') = \begin{cases} 1 & \text{if } v' = v \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

  In other words, there exists a single request for each location in the graph that can be satisfied at any time but only in its desired location.

- The list A with only one agent defined by $(v_i, 0, v_i, l + 1, c(*) = 0)$. In other words, the agent starts moving from any location v_i at the first time step and is expected to arrive back at to its start location v_i at some time

[1] The cost of a path p_j can be defined as the extra number of hops in that path when compared to the shortest path that can be used for the journey (as implemented later in this paper), or it can be defined as the difference between the agent's latest time of arrival t_f and the actual time of arrival.

before $t_f = l+1$, which is enough time for the agent to visit every location in the graph only once. Moreover, the cost function of the agent is defined to be 0 for any path chosen.

To find the optimal solution for this instance of the GRS problem, we need to find a path for the agent which goes through each node in the graph to satisfy all available requests thus maximuming the total revenue. Moreover, each node in the graph has to be visited only once to allow the agent to return to his start location at the defined time $t_f = l + 1$. Finding such a path amounts to finding a Hamiltonian cycle in the graph starting and ending at the same node v_i, thus reducing our problem to the Hamiltonian Cycle problem. Since we know that the problem of finding a Hamiltonian cycle in a graph is NP-Complete [15], we prove that the GRS problem is also NP-Complete. ■

3 Geo-Temporal Request Satisfaction Games

Due to the NP-Complete nature of the GRS problem, we develop a heuristic mechanism that takes into consideration the rationality of the agents, and recasts the problem in a game-theoretic setting. We define the rules of the mechanism that allow agents to compete for servicing the highest paying requests in the field. The objective of the game is to achieve a total profit that is as close as possible to the social optimal.

3.1 Geo-temporal Request Satisfaction Game

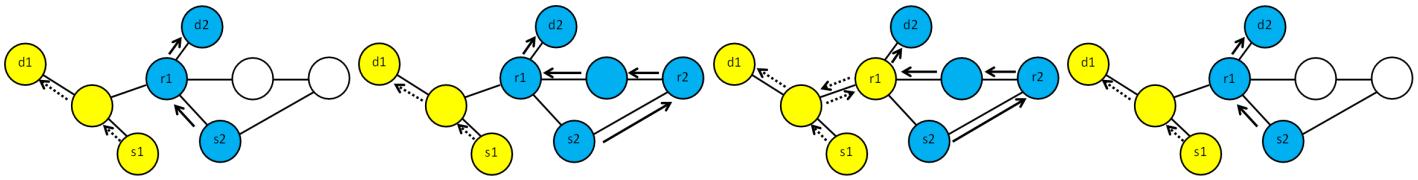
As mentioned above, we recast the GRS problem as a game in which the each agent represents a single player. In such a game, we define a player's better response move as a proposal to take a new path that improves its utility, given that a player has knowledge about the current state of the system (e.g., the requests available and other players moves). Moreoever, players moves are assumed atomic and serial.

Definition 3 *(The GRS Game) In the geo-temporal request satisfaction game, players take turns in making better response moves that maximize their utility until all players are satisfied with their path choices. The utility of a player in the GRS game is defined as the total profit provided by the path chosen,*

$$U_{x_i}(p_j) = \sum_{r_k \in p_j} (p_{r_k}()) - c(p_j) \tag{2}$$

where p_j is a legitimate path chosen by player x_i, $p_{r_k}()$ is the valuation function of request r_k and $c(p_j)$ is the cost of path p_j as defined by the cost function of the agent.

Definition 4 *(Domination Rule) Since players may choose to service the same request, we define an arbitiration rule that decides which player is allowed to service the request. Namely, the* Domination Rule *states that a player x_i is*



(a) Initial state (b) State after first move (c) State after second move (d) State after third move

Figure 1: Counter example that proves the non-convergence of the GRS game.

allowed to dominate another player x_j and claim a request r_k serviced originally by x_j if the total profit obtained from r_k when serviced by player x_i is strictly higher that when serviced by player x_j.

Theorem 5 *The GRS game may never reach Nash Equilibrium under better response dynamics.*

Proof. We prove this by providing a counter-example with an instance of the problem, in which players never stop making the same set of repeated moves. Consider the graph shown in the Fig.1, the set $A = (s1, 1, d1, 4, c(*) = 0), (s2, 1, d2, 6, c(*) = 0)$ of two players with zero cost functions, and the set $R = (r1, 2, p_1()), (r2, 2, p_2())$ of two requests with $p_1(v, t) = 7 - (t - 2)$ if $v = 5, t \geq 2$ and 0 otherwise and $p_2(v, t) = 3$ if $v = 7, t = 2$ and 0 otherwise.

Assume the initial state of the game shown in Fig.1a, a_2 claims request r_1 for a utility of 7 while a_1 has no choice but to stick to its shortest path with a utility of 0. For the first move, a_2 decides to change its path and claims both requests r_2 and r_1 in that order with a utility of 8, giving a_1 the chance to dominate it and claim r_1 with a utility of 6. Now that a_2 has lost r_1 and has a utility of only 3, it makes a move and changes its path again and decides to claim r_1 only for a utility of 7, and a_1 again has no choice but its shortest path with a utility of 0. This sequence of moves is repeated over and over again, leading to the non-convergence of this instance of the game. Thus, proving that the GRS game does not always converge under better response dynamics.

■

3.2 Multi-stage Request Satisfaction Game

The dependability of the player's utility function in the GRS game on the total valuation of all requests serviced on the path chosen is the reason the game does not always converge under better response dynamics. We overcome this problem by redefining the player's utility function to depend on the valuation of the single highest paying requests that can be serviced on the path chosen.

Definition 6 *(The GRS_1 game) The modified geo-temporal request satisfaction game takes as input the set of available requests and the set of agents journeys*

and cost functions. Players take turns in making better response moves that maximize their utility until all players are satisfied with their path choices, and the domination rule is applied. The utility of a player in the GRS game is defined as,

$$U_{x_i}(p_j) = max(p_{r_k \in p_j}()) - c(p_j) \tag{3}$$

where p_j is a legitimate path chosen by player x_i, $p_{r_k}()$ is the valuation function of request r_k and $c(p_j)$ is the cost of path p_j as defined by the cost function of the agent.

We prove below that the GRS_1 game does converge for all instances of the game, for an extended proof please refer to our technical report [5]. After convergence, each player decides on the path with the highest paying request, and marks the request's exact location and time as part of its journey. In other words, the player's original journey is divided into two smaller journeys; the first new journey starts at the same location and time as the original journey and ends at the marked location and time of the highest paying request, and the second journey starts from the marked location and time and ends at the original journey's destination location and time.

Theorem 7 *The GRS_1 game reaches Nash Equilibrium under better response dynamics.*

Proof. To prove this theorem we show that the GRS_1 game is an exact potential game [28] with an increasing potential function,

$$\Phi(s_i, s_{-i}) = \sum_{r_k \in R} (p_{r_k} - cost(x_i)) \tag{4}$$

where p_{r_k} is the valuation function of the request r_k and $cost(x_i)$ is the cost incurred by the player x_i that serviced that request r_k.

In other words, the function $\Phi(s_i, s_{-i})$ measures the total profit of the system after a player x_i makes a move to the state of s_i. According to the definitions of the potential function of the system and the utility function of the players, we guarantee that the potential function $\Phi(s_i, s_{-i})$ is always increasing. Moreover, since we know the maximum valuations that can be obtained from all requests available and the maximum costs incurred by each player in the game, we know that there exists a maximum profit value that the function $\Phi(s_i, s_{-i})$ cannot exceed. Therefore, we guarantee that the GRS_1 game reaches Nash Equilibrium under better response dynamics.

■

According to the definition of the GRS_1 game, at most a single request can be serviced by each player. As a result, there may be requests that are left unserviced. To service these leftover requests, we allow the agents to repeat the game in a *divide-and-conquer* approach.

Definition 8 *(The GRS_1^* game) In the multi-stage geo-temporal request satisfaction game, the GRS_1 game defined above is repeated in a d̈ivide-and-conqueräpproach. In the first stage, the input of the GRS_1^1 game is the set of all available requests, and the set of agents journeys and cost functions. Then, for each stage k, the input of the GRS_1^k game is the set of leftover requests and the set of new journeys and cost functions obtained from the output of the GRS_1^{k-1} game. The multi-stage game stops when the output of the GRS_1^{k+1} game is the same as that of the GRS_1^k game,* i.e.*, no more requests can be satisfied.*

Lemma 9 *The number of stages in a GRS_1^* game is polynomial.*

Proof. In each GRS_1 game, at most a single request can be serviced by each player. Therefore, at the end of each stage of the GRS_1^* game one of these three scenarios can occur, 1) No requests are satisfied, so the whole GRS_1^* game ends, 2) Only a single request is satisfied from a total of n requests, thus leaving $n-1$ requests to be satisfied in the next stage, 3) More than one request (i) is satisfied from a total of n requests, thus leaving $n-i$ requests to be satisfied in the next stage. From these three scenarios, the worst case scenario is the second one, and if it occurs in each stage of the GRS_1^* game, we will get a total number of $|R|$ stages which is a polynomial. ■

Theorem 10 *The Multi-step Request Satisfaction game reaches equilibrium under better response dynamics.*

Proof. By combining our conclusions from Theorem 7 and Lemma 9, we prove that the GRS_1^* game has a polynomial number of stages and that the GRS_1 game played in each stage always converges to a Nash equilibrium. Therefore, any instance of the GRS_1^* game is proven to always converge under better response dynamics. ■

4 Performance Evaluation

To evaluate the performance of the GRS_1^* game, we designed several sets of simulation-based experiments to compare it to other approaches and under different simulation settings. In this section, we explain our simulation setting and the results obtained from the experiments performed on the GRS_1^* game.

4.1 Simulation Setting

We developed a C-sharp-based simulator to evaluate the performance of the GRS_1^* game under different settings. The input of each simulation is a graph representing the mobility field, the request arrival rate, the agent arrival rate and the total simulation time. Once the simulation starts, a list of requests and agents are generated with random attributes. Each request is assigned a uniformly random location and time, and for simplicity, all requests are defined with the same valuation function. Each agent is assigned a random schedule

with uniformly random start location, start time, destination, latest arrival time, and for simplicity, all agents are defined with the same cost function. The cost function used by all agents in the simulator is defined as the extra number of hops in the path chosen when compared to the shortest path that can be used for the agent's journey.

After all requests and agents are generated, the initial stage of the GRS_1^* game starts, and the initial list of players is created. Players have a set of attributes that do not change during the stage, which are the start location, start time, destination, latest arrival time and the set of possible paths from the start location to the destination. Since the process of computing the list of all possible paths between the start location and the destination is exponential in nature, we use Yen's ranking algorithm [27] to calculate the set of k-shortest paths from the player's start location to his destination. Then, we ensure that only legitimate paths – the paths with suitable lengths – are added to the player's set of possible paths.

Each player also has a set of decision variables that are used during the GRS_1^k game played at each stage k to define its moves; the location it decides to visit, the time it decides to visit it at, and the utility expected from such a move. In each GRS_1^k game, new players with journeys obtained from the output of the GRS_1^{k-1} game join the game, and take turns to make better response moves that maximize their utlity. The GRS_1^k game ends when no player wishes to make another move, thus ending a stage. Then, all satisfied requests are removed from the requests' list and a new players' list is created as described in the previous section. This is repeated until no more requests can be satisfied.

When it's a player's turn to decide whether to make a move or not, it first checks whether it has been dominated by another player and resets its decision variables if true. Then, the player picks a random path from its set of possible paths to the destination, and calculates the utility of that path as defined in the GRS_1 game. If the highest paying request was claimed by another dominating player, the original player has to search for the request with the second highest profit on the same path, and update the path's utility accordingly. This process is repeated till either the player finds a suitable request, or no requests are available. If an unclaimed request is found, and it provides a higher utility value for the player, the player decides to make a move, selects the chosen path, claims the request and updates its decision variables. Otherwise, the player decides not to make a move.

4.2 Simulation Results

We created several sets of experiments to evaluate the MRS game's performance, and their results are shown below. In all the experiments, we define the *profit ratio* as the evaluation metric, which is the ratio of the total profit obtained from the game, to the maximum profit available. Moreover, for each set of experiments, we perform 25 simulations and calculate the average profit ratio.

In the first set of experiments, we compare the performance of the game to

Algorithm 1 Algorithm which describes how MRS is implemented.

```
stopGame ⇐ False
lstPlayers ⇐ Agents
while NOT(stopGame) do
  stopGame ⇐ True
  stopStep ⇐ False
  while NOT(stopStep) do
    stopStep ⇐ True
    for all p ∈ lstPlayers do
      if p makes a move then
        stopStep ⇐ False
      end if
    end for
  end while
  lstNewPlayers ⇐ Empty
  for all p ∈ lstPlayers do
    if p.decRequest ≠ null then
      stopGame ⇐ False
      lstNewPlayers.ADD(Player(p.startLoc, p.startTime, p.decLoc,
      p.decTime, p.decTime-p.startTime))
      lstNewPlayers.ADD(Player(p.decLoc, p.decTime, p.finishLoc,
      p.finishTime, minLength(p.decLoc,p.finishLoc)))
    end if
  end for
  lstPlayers = lstNewPlayers
end while
```

Algorithm 2 Algorithm which descibes how a player decides to make a move.

```
if decRequest ≠ null and decRequest.satAgent ≠ thisID then
    utility ⇐ 0
    decLoc ⇐ finishLoc
    decTime ⇐ finishTime
    decRequest ⇐ null
end if
Pick a path from the k-possible paths at random
Get highest paying unclaimed request (r) on that path
Calculate the newUtility of that path
if newUtility ¿ utility then
    utility ⇐ newUtility
    decLoc ⇐ r.location
    decTime ⇐ r.time
    decRequest ⇐ r
    r.profit ⇐ newUtility
    r.satAgent ⇐ thisID
    return TRUE
end if
return FALSE
```

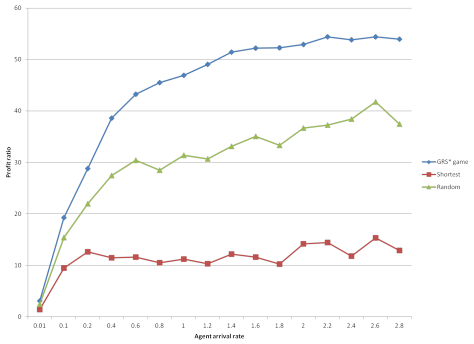


Figure 2: The comparison between the MRS game, shortest path and random path approachs.

two traditional approaches for request satisfcation in geo-temporal settings, the shortest path and the random path approaches. In the shortest path approach, agents choose the shortest path available to go to their destinations, and in the random path approach agents choose any random legitimate path. In both approachs, all the requests in the way of the agents are serviced. This set of experiments is based on a $200 * 200$ cartesian Manhatten-style grid, with a request arrival rate of 0.4 requests per time unit and total simulation time of 1000 time units.

The results shown in Fig.2 represent the average profit ratio of all ap-

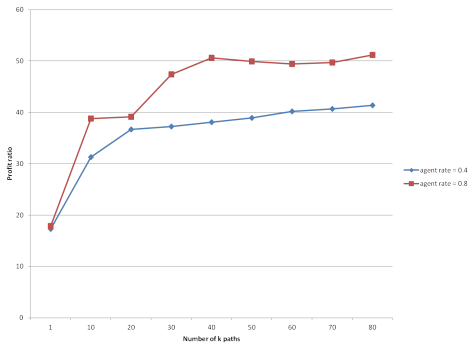


Figure 3: The effect of increasing the number of k-shortest paths used by each player.

proaches, when varying the agent arrival rate from 0.01 to 2.6 agents per time unit. The results show the superiority of the GRS_1^* game; the performance improves monotonically as the number of agents in the field increases. Although the performance of the game improves as the agent arrival rate increases, the experiments shows that the marginal utility of considering additional agents is negligible for higher values of the agent arrival rate, which shows that the game performs as expected. As for the other approaches, the performance of the shortest path approach is always less than 15% and irregular, while the performance of the random approach becomes irregular after an agent arrival rate of 0.6.

In the second and third sets of experiments, we evaluate the GRS_1^* game's performance under different settings. The second set of experiments is based on a $200 * 200$ cartesian Manhatten-style grid with a request arrival rate of 0.4, and a total simulation time of 1000 time units. We vary the number k of shortest paths obtained from Yen's ranking algorithm to be used in the set of possible paths to the destinations from 1 path - the shortest - to 80 paths. The results shown in Fig.3 represent the average profit ratio for agent arrival rates of 0.4 and 0.8. The results show that only a small number of k – smaller state space – provides us with almost maximal performance. In other words, the marginal utility of considering additional paths is negligibal with larger values of k, which also shows that the game performs as expected.

The third set of experiments is based on a $200 * 200$ cartesian Manhatten-style grid and a $15 * 15$ cartesian Manhatten-style grid with a request arrival rate of 0.4 and a total simulation time of 1000 time units. The results shown in Fig.4 represent the average profit ratio when varying the agent arrival rate from 0.01 to 2.8 agents per time unit. These results show that an increase in the agents density on the field, improves the game's performance, proving its accuarcy.

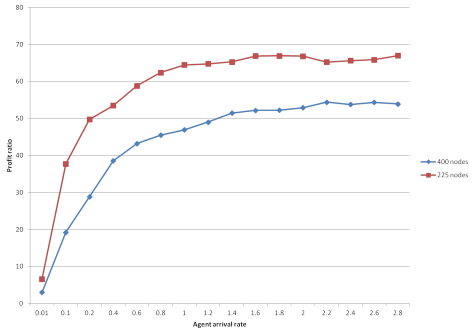


Figure 4: The effect of increasing the agent's density in different sized graphs.

Table 1: Our classification of geo-presence-capable systems.

Mobility pattern of agents	**Example**
Stationary	Wireless sensor networks [23]
Controlled mobility	Robotics [42] Mobile wireless sensor networks [11]
Coordinated mobility	Work done in [30]
Uncontrolled mobility	Opportunistic sensor networks [8] Crowdwoursing applications [4]

5 Related Work

The resource allocation problem is a classic problem that has been studied for several years in distributed systems such as in [14, 32] where resources are reserved to satisfy the requests in the system to improve system utilization while providing a quality of service guarantee for the requests. In [9], authors propose algorithmic mechanisms that maximizes the benefit of the system while providing a guarantee of correctness and reliability of the resources reserved for request satisfaction. For crowdsourcing-based geo-presence capable systems, such classic resource allocation algorithms will not be appropriate due to the mobile nature of the agents in the system which are considered the resources needed to be allocated for the requests. Our classification of the four different classes of geo-presence-capable systems according to the mobility patterns of the agents is shown in Table 1.

Traditional wireless sensor networks as described in [23, 3, 2, 10] are direct examples of the first class of geo-prescence-capable systems that are based on stationary agents. In such systems, the sensors are set up in fixed geographical locations in the field and are viewed as the stationary agents of the system, sensors continuously collect and process data and send them to aggregate points to be furthur processed for specific applications. Data from wireless sensor networks are mainly processed using two different methods, online and offline.

In the online method, the data collected by the sensors are processed in real-time for specific application requirements such as in [40, 26]. In the offline method, the data collected by the sensors are stored in databases to be queried by users in the future, in this case the spatio-temporal request satisfaction process is defined as the transformation of the requests into an appropriate queries to be applied on the database. Most work in spatio-temporal databases has been on defining an appropriate query language, improving query processing and improving data collection from sensor nodes as in [6, 16, 18].

In the second class of geo-presence-capable systems, agents in the system have a controlled mobility pattern which means that their mobility is controlled by the system itself. Agents in these systems are usually robots or mobile sensor nodes with no selfish motive and their path decisions are made to accomplish the system's objective. Mobility control is widely used for field coverage [19, 42, 45], maintainance of communication chains [33, 11] or for specific task accomplishment [35, 34, 31, 29, 17] and they vary from centralized to decentralized approaches according to the requirements and the assumptions of the system.

In the third class of geo-presence-capable systems, agents have uncontrolled mobility patterns. Such agents are self-motivated with predefined schedules. They willingly participate in the system and decide whether or not to perform a task - service a request - according to their prior plans and they may alter their schedules to perform a task if given the right incentive to do so. However, that decision is solely theirs and the system cannot dictate and/or predict their behaviour. In this third class of geo-presence capable systems, clients are not given any guarantees about the response time or response quality for their requests and the request satisfaction process is an ad hoc, opportunistic process. Crowdsourcing systems fall into this class of systems whether enterprise-based crowdsourcing applications as Amazon Mechanical Turk [4] or location aware systems as in [1] and [44].

In enterprise-based crowdsourcing systems such as Amazon Mechanical Turk [4], Flicker [43], Poptent [36] and UTest [41], clients simply post their requests and state what needs to be done and the payment they are willing to pay for satisfying the specified request. Agents access the requests list and decide which requests they can satisfy according to the nature of the request and the agent's schedule. Research on such systems ranges from analysis of the different system components and the quality of work performed by the agents [20], to proposals of how task description and pricing should be standardized to achieve better service and better guarantees [21, 39], to work on how to provide incentives for agents to participate in such crowdsourcing systems [24]. Due to the nature of the agents in such systems and their method of accessing the requests, the request satisfaction process depends solely on the decisions of the agents and the system cannot provide any guarantees to its clients.

For location-aware crowdsourcing systems, researchers have been recently proposing to incorporate human hand-held devices into sensor networks thus creating what is known as opportunistic sensor networks. In this variation of sensor networks, the sensor nodes are the devices carried by the mobile agents,

such as smart phones, laptops or special sensor equipped vehicles. Authors in [1, 44, 22] propose visions of building a complete heterogenous opportunistic sensor network which can be used for any crowdsourcing application, while authors in [25] propose and build a specific crowdsourcing applicaton for only image translation and authors in [7] propose and build a Twitter-based application for answering location-based queries. Work done on opportunistic networks focuses on how to improve the quality of feedback in the system by either allowing agents to alter their transmission range radii to cover more area and satisfy more requests while applying a rendezvous opportunities to exchange information about requests between the agents while minimizing energy consumption [13, 12], or by allowing agents to delegate work to each other [8] , or by allowing all the request scheduling process to be performed in a distributive manner where requests are sent from one agent to the other till the agent with the right requirements recieve it and satisfy it [38]. As mentioned above, the request satisfaction process in such systems is opportunistic and ad hoc, however, recent systems such as in [39, 25, 7, 37] propose to assign a request to a set of agents who have a matching context with it.

Finally we define the last class of geo-presence-capable applications, where agents have coordinated mobility patterns. Agents with coordinated mobility patterns are regular self-motivated mobile agents as explained above ,but are willing to do minor modifications to their schedules given the right incentive. This notion of mobility coordination has first been proposed in [30] according to our knowledge where the authors assume that mobile nodes have a flexibility in their schedule and they leverage this flexibility or what they define as slack to obtain a certain coverage distribution of the network.

6 Conclusion and Future Work

In this paper we defined the problem of request satisfaction in geo-presence-capable crowdsourcing applications and proved it to be NP-Complete. We proposed a practical solution mechanism for the problem, namely the GRS game, and proved that it might not always converge under better response dynamics because of the definition of the players utility functions. We then proposed a modified version of the game, namely the GRS_1^* game, and proved that it always converges under better response dynamics, which we then evaluated using simulated experiments, and showed its accuracy and superiority.

We are currently building a prototype for the proposed GPaaS framework, in which the GRS_1^* game is used for request satisfaction. We are implementing the main components of the framework on a cloud-based platform to provide the framework's services to clients and agents using any device from any location. Clients and agents can access these services through traditional web-based browsers, or through mobile applications that we are currently developing. Moreover, we intend to improve the method by which a player chooses a path when playing the GRS_1^* game. The current method of choosing a path from the set of possible paths obtained by Yen's ranking algorithm is computationaly

expensive, thus affecting the game's total response time.

Acknowledgements

This research was supported by NSF awards #1012798, #0952145, #0820138, #0720604, and #0735974.

References

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, pages 20–29, 2007.

[2] I. Akyildiz. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, Mar. 2002.

[3] I. Akyildiz and W. Su. ACCEPTED FROM OPEN CALL-A SURVEY ON SENSOR NETWORKS-Recent advancement in wireless communications and electronics has enabled. *IEEE*, (August):102–114, 2002.

[4] Amazon Inc. Amazom Mechanical Turk.

[5] C. Bassem and A. Bestavros. Mechanism design for spatio-temporal request satisfaction in mobile networks. Technical report, Boston University, 2011.

[6] P. Bonnet, J. Gehrke, and P. Seshadri. Towards sensor database systems. In *Mobile Data Management*, pages 3–14. Springer, 2001.

[7] M. F. Bulut. Crowdsourcing Location-based Queries. *Architecture*, pages 513–518, 2011.

[8] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. a. Peterson. People-centric urban sensing. *Proceedings of the 2nd annual international workshop on Wireless internet - WICON '06*, pages 18–es, 2006.

[9] E. Christoforou, A. Fernández Anta, C. Georgiou, and M. Mosteiro. Brief Announcement: Algorithmic Mechanisms for Internet-Based Computing under Unreliable Communication. *Distributed Computing*, 0609:147–149, 2011.

[10] D. CRULLER and D. Estrin. Overview of sensor networks. *Computer*, (August):41–49, 2004.

[11] C. Dixon and E. W. Frew. Maintaining Optimal Communication Chains in Robotic Sensor Networks using Mobility Control. *Mobile Networks and Applications*, 14(3):281–291, Sept. 2008.

[12] S. Eisenman, N. Lane, and A. Campbell. Techniques for improving opportunistic sensor networking performance. *Distributed Computing in Sensor Systems*, pages 157–175, 2008.

[13] S. B. Eisenman, H. Lu, and A. T. Campbell. Halo : Managing Node Rendezvous in Opportunistic Sensor Networks. *New York*.

[14] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advance reservations and co-allocation. In *1999 Seventh International Workshop on Quality of Service. IWQoS'99. (Cat. No.98EX354)*, number 1, pages 27–36. Ieee, 1999.

[15] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, volume 54, pages 47–63. ACM, 1974.

[16] R. Govindan, J. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker. The sensor network as a database. Technical report, Citeseer, 2002.

[17] K. Gupta and Z. Yao. Distributed Strategies for Local Minima Escape in Motion Planning for Mobile Networks. *Proceedings of the 2nd International Conference on Robotic Communication and Coordination*, 2009.

[18] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *Network, IEEE*, 18(1):6–14, 2004.

[19] A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. *Distributed autonomous robotic systems*, 5:299–308, 2002.

[20] P. Ipeirotis. Analyzing the amazon mechanical turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[21] P. Ipeirotis and J. Horton. The Need for Standardization in Crowdsourcing. 2011.

[22] A. Kansal, S. Nath, J. Liu, and W. I. Grosky. SenseWeb : An Infrastructure for Shared Sensing. *Earth*, pages 8–13, 2007.

[23] S. Kumar. Sensor networks: Evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256, Aug. 2003.

[24] Y. Liu, V. Lehdonvirta, T. Alexandrova, M. Liu, and T. Nakajima. Engaging Social Medias: Case Mobile Crowdsourcing. *Search*, pages 1–4, 2011.

[25] Y. Liu, V. Lehdonvirta, M. Kleppe, T. Alexandrova, H. Kimura, and T. Nakajima. A crowdsourcing based mobile image translation and knowledge sharing service. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, page 6. ACM, 2010.

[26] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02*, page 88, 2002.

[27] 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, 2003.

[28] D. Monderer and L. S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124 – 143, 1996.

[29] V. Montreuil, A. Clodic, M. Ransan, and R. Alami. Planning human centered robot activities. *2007 IEEE International Conference on Systems, Man and Cybernetics*, pages 2618–2623, Oct. 2007.

[30] H. Morcos and A. Bestavros. Preferential field coverage through detour-based mobility coordination. *Workshop (Med-Hoc-Net), 2010 The*, 2010.

[31] Y. Mostofi. Decentralized Communication-Aware Motion Planning in Mobile Networks: An Information-Gain Approach. *Journal of Intelligent and Robotic Systems*, 56(1-2):233–256, May 2009.

[32] M. Netto, K. Bubendorfer, and R. Buyya. SLA-based advance reservations with flexible and adaptive time QoS parameters. *Service-Oriented ComputingICSOC 2007*, pages 119–131, 2007.

[33] A. Oda and T. Nakabe. Quality of connectivity guarantee of ZigBee based wireless mobile sensor network. *Informatics, 2009. INDIN 2009. 7th IEEE*, pages 3–8, 2009.

[34] T. Okada, R. Beuran, J. Nakata, Y. Tan, and Y. Shinoda. Collaborative motion planning of autonomous robots. *2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007)*, pages 328–335, Nov. 2007.

[35] D. R. Parhi, S. K. Pradhan, A. K. Panda, and R. K. Behera. The stable and precise motion control for multiple mobile robots. *Applied Soft Computing*, 9(2):477–487, Mar. 2009.

[36] Poptent. Poptent, 2011.

[37] H. Psaier, F. Skopik, D. Schall, and S. Dustdar. Resource and Agreement Management in Dynamic Crowdcomputing Environments. *infosys.tuwien.ac.at*.

[38] O. Riva and C. Borcea. The Urbanet Revolution :. In *Networks*, pages 41–49.

[39] J. Sousa, V. Poladian, D. Garlan, B. Schmerl, and M. Shaw. Task-based adaptation for ubiquitous computing. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 36(3):328–340, May 2006.

[40] J. Stankovic, Q. Cao, and T. Doan. Wireless sensor networks for in-home healthcare: Potential and challenges. *High Confidence*, pages 7–10, 2005.

[41] uTest Inc. Software Testing: UTest, 2011.

[42] G. Wang, G. Cao, and T. F. La Porta. Movement-assisted sensor deployment. *IEEE Transactions on Mobile Computing*, 5:640–652, June 2006.

[43] Yahoo! Inc. Flicker: Creating Commons, 2011.

[44] F. Zambonelli. Pervasive urban crowdsourcing: Visions and challenges. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 578–583. IEEE, 2011.

[45] Y. Zou and K. Chakrabarty. Sensor Deployment and Target Localization Based on Virtual Forces. *Computer*, 00(C), 2003.