TRACCS: Trajectory-Aware Coordinated Urban Crowd-Sourcing

Cen Chen and **Shih-Fen Cheng** and **Aldy Gunawan** and **Archan Misra**
School of Information Systems
Singapore Management University

Koustuv Dasgupta and **Deepthi Chander**
Xerox Research Centre India

Abstract

We investigate the problem of large-scale mobile crowd-tasking, where a large pool of citizen crowd-workers are used to perform a variety of location-specific urban logistics tasks. Current approaches to such mobile crowd-tasking are very decentralized: a crowd-tasking platform usually provides each worker a set of available tasks close to the worker's current location; each worker then independently chooses which tasks she wants to accept and perform. In contrast, we propose *TRACCS*, a more coordinated task assignment approach, where the crowd-tasking platform assigns a sequence of tasks to each worker, taking into account their expected location trajectory over a wider time horizon, as opposed to just instantaneous location. We formulate such task assignment as an optimization problem, that seeks to maximize the total payoff from all assigned tasks, subject to a maximum bound on the detour (from the expected path) that a worker will experience to complete her assigned tasks. We develop credible computationally-efficient heuristics to address this optimization problem (whose exact solution requires solving a complex integer linear program), and show, via simulations with realistic topologies and commuting patterns, that a specific heuristic (called *Greedy-ILS*) increases the fraction of assigned tasks by more than 20%, and reduces the average detour overhead by more than 60%, compared to the current decentralized approach.

Introduction

Large-scale mobile crowd-tasking, where citizen volunteers are incentivized to perform location-specific tasks, has recently attracted strong commercial interest. Several companies/business visionaries now believe in the potential of such a *urban crowd logistics* paradigm, where a participative pool of urban "crowd-workers" are co-opted to perform (typically in exchange for micro-payment based incentives) a variety of last-mile tasks, such as performing frequent checks of the display shelves at neighborhood stores, delivering packages for consumers and reporting on the queuing wait times at various restaurants and food courts. State-of-the-art mobile crowd-tasking solutions (e.g., FieldAgent, GigWalk) largely exhibit two characteristics: (a) they focus on a *pull-based, proximity-driven* model of task selection, where each crowd-worker selects tasks from a set of available tasks that are near the worker's *current location*, and (b) the task selection process is inherently *decentralized*, with each worker opportunistically selecting one or more tasks that are most appealing to her.

Absent from such approaches are the fundamental concepts of (i) *Large-scale Task Coordination*, where the crowd-tasking platform proactively recommends tasks to each crowd-worker in a globally coordinated way, with the objective of improving the acceptance and completion rate of such tasks, and (ii) *Predictive Crowd-Tasking*, where the recommendation or selection of tasks is not performed myopically (based on just the current location context of the workers), but instead exploits an individual's *movement trajectory* over a longer time horizon (such as an entire day). We believe that adoption of these two concepts can significantly improve the efficacy of the crowd-tasking process:

- By providing better selection of tasks across tens of thousands of urban crowd-workers, centralized spatiotemporally-aware coordination is likely to increase the overall task acceptance and completion rate. For example, that given two tasks T_1 and T_2, of which T_1 lies very close to the travel routes of both Alice and Bob, while T_2 involves a short detour for Alice but is very far from Bob's travel path, the platform should recommend task T_1 to Bob instead of Alice, as this would free Alice up to perform task T_2. Current crowd-tasking platforms fail to take advantage of such globally advantageous possibilities—e.g., *FieldAgent* displays tasks around the current location of the worker, providing the worker with no additional guidance on what tasks to select.
- Suggesting tasks, based on a longer time horizon of movement trajectory (as opposed to instantaneous location), is likely to increase the overall payoff for each individual crowd-worker, by making the best opportunistic use of *slack time* during the day. For example, if the crowd-tasking platform was aware of the commuting pattern of a crowd-worker, it might recommend her multiple tasks that are in vicinity of the two different bus stops where she has to spend 6-7 minutes each waiting for bus transfers, and which she might conceivably complete during those waiting periods. In a purely pull-based mechanism, the worker may not have been aware of these opportunities.

In this paper, we shall investigate such a ***Tr**ajectory-**A**ware **C**entrally-coordinated **C**rowd-**S**ourcing* framework, called *TRACCS*[1], where the tasks proactively recommended by the tasking platform take into account the predicted trajectory (movement pattern) of the individuals. *TRACCS* is motivated by the increasing availability of individual-specific movement traces, both outdoors and indoors, captured by technologies such as GPS, Wi-Fi based location tracking and inertial sensing on mobile devices. Such location traces, combined with trajectory prediction/analysis techniques, now makes it possible to predict the likely "regular" travel pattern of each individual–e.g., we can now automatically infer that Alice will leave her office in City Center at approximately 5:30 on Thursday evening, then head by bus to the book club gathering at her club at 6:30, then take a train to the gym (located 6 blocks from her house) around 8pm, before finally walking home along First Avenue around 9pm.

To develop this framework, we shall formulate task assignment as a constrained optimization problem, whose goal is to maximize the total *reward* earned by all crowd-workers over a specified time horizon–i.e., effectively maximize the total amount of payoff resulting from all tasks *completed* as per the task-specific requirements. To accommodate the overheads of task execution, the optimization is subject to a set of individual worker-specific *detour constraints*–expressed as an upper bound on the maximum additional time (or distance) that a worker can spend in completing her assigned tasks, over and above that spent on her normal movement trajectory. This is based on the typical case, where the worker's regular trajectory is defined by her customary (or *routine*) commuting pattern, and the completion of a location-specific task (e.g., checking for the availability of a product at the convenience store located near her) requires her to deviate from her usual route (e.g., the walk back from the gym to her home). We can generalize this optimization problem to accommodate a variety of real-world inspired attributes for the tasks, such as *differing coverage requirements* (how many duplicate workers are needed till an appropriate level of task fidelity is reached?) or *priority* (given limited workers, which tasks should be preferentially assigned over others). While *TRACCS* may be viewed as part of the general class of *orienteering problems* (Vansteenwegen, Souffriau, and Oudheusden 2011), developing a practical solution requires us to tackle several important challenges.

Accordingly, our **Key Challenges & Contributions** are:

1. *Capturing Diverse Optimization Objectives Precisely:* While the overall optimization objective is constant—maximizing the sum of the rewards for all assigned tasks–different movement patterns can impose different types of detour constraints. To capture such diversity precisely, we present two different integer linear program (ILP)-based optimization models, that *precisely* solve the problem for two separate scenarios: a *detour-without-retrace* model, where the worker simply moves to the closest downstream point in her yet-to-be completed routine trajectory after completing a task-driven detour (this represents the case where Alice takes a detour during her walk home), and a *detour-with-rejoin* model, where the worker must return to the point on the routine trajectory where she started her detour and resume the yet-to-be completed portion (this represents that case where Alice must return to the bus interchange to resume her journey after performing a task while waiting there).
2. *Handle City-Scale Worker Populations and Tasks:* The ILP-based formulations have exponential computation complexity, and prove to be intractable as the number of workers (or tasks) grows modestly, even to a few tens. To make centralized task assignment feasible, even for tens of thousands of workers and/or tasks, we define and evaluate multiple *assignment heuristics* (for both the "detour-without-retrace" and "detour-with-rejoin" models), and demonstrate that our heuristics provide good-quality and fast task assignment (e.g., in 2 seconds, given a set of 1000 workers and 2000 tasks, it can devise a solution that assigns 98.6% of the total tasks).
3. *Practical Benefits in Realistic Situations:* We are currently working towards a real-world deployment and testing of *TRACCS* –this is of course, a formidable practical challenge. Meanwhile, to *realistically* quantify the expected performance benefit, we test our Heuristic solutions on large-scale synthetic movement traces (e.g., 1000+ workers and 5,000+ tasks), generated over trajectories that mimic daily commuting patterns (including intermediate transfer points) from one or more residential areas to a central business district. Our results show that: (i) our Heuristic solutions scale quite well, in terms of both computational complexity and assignment quality (computing, in less than 0.3 seconds, assignments that cover over 85% of the tasks) and (ii) by optimizing over a longer time horizon (1 day), *TRACCS* can significantly outperform decentralized, instantaneous proximity-based task selection (with $\sim$20% or higher increase in the number of assigned tasks, and a $\sim$60% reduction in the average detour overhead).

[1]Pronounced as "tracks"

Related Work

Given the large amount of literature on crowd-sourcing in general, we focus on describing the key prior results related to the distinct themes of our research agenda.

Location-Aware Crowd-Tasking: Current mobile crowd-sourcing platforms, such as FieldAgent, GigWalk, NeighborFavor and mCrowd, utilize the current location of a worker to provide her with a set of geographically proximate tasks. The worker is permitted to pull tasks that they can perform, but has to figure out how to maximize the earnings, even while amortizing travel costs and adhering to the task deadlines. Kazemi and Shahabi provides a framework whereby a crowd-sourcing server can prioritize the tasks assigned to available workers using knowledge about the worker's current availability and location. More specifically, the workers in the vicinity of a specific task's location are queried to determine their willingness to accept the task; however, different strategies may be used to determine the

order in which workers are queried (e.g. greedy, low location entropy, or nearest neighbor priority). In all these mobile micro-tasking environments, the onus of making the choice of tasks and sequencing them so as to maximize utility to the worker while ensuring task completion, lies solely with the worker.

Optimizing Individual-Level Task Selection: There have been a few influential studies that investigate how individual workers go about selecting tasks, when presented with visibility into the location-dependence of these tasks. Using data from a real-world mobile crowd-tasking platform, Musthag and Ganesan (2013) show how a very small set (less than 10%) of workers, called *super-agents*, perform the vast majority of such tasks; these super-agents experience travel detours that are more than double that of the rest of the worker population, and are also adept at sequencing tasks efficiently to maximize their rewards. Clearly, the job of selecting the best *sequence* of tasks, taking into account complex factors such as expected travel times and task deadlines is a very non-trivial one, as evidenced by the poor success rate of 90% of the worker pool. Additionally, prior studies (e.g., see Kokkalis et al. (2013) and Talamadupula et al. (2013)) also show that people tend to complete their tasks better when they are provided with action plans (by automated planning tools) to perform their tasks. These results provide strong implicit evidence for the promise of our *TRACCS* framework. If a sequence of tasks could be automatically recommended, and if such tasks were chosen so as to impose minimal additional overhead (in terms of detours) from an individual's lifestyle-driven movement pattern, then it is very likely that a larger fraction of the worker population would perform a progressively larger set of the tasks.

Predicting Movement Trajectories: There has recently been an explosion of research on both sensing and predicting individual user movement, both outdoors and indoors. In outdoor environments, such location trajectories can be obtained and analyzed (e.g., Becker et al. (2013)) using cellular data available to telecommunications companies. Alternately, specialized location tracking technologies (e.g., PlaceLab (LaMarca et al. 2005)) have also been proposed for ubiquitous location and movement tracking. Recent efforts have focused on obtaining large-scale indoor movement traces–e.g., the LiveLabs testbed (Misra and Balan 2013) is being progressively deployed to track the trajectories of thousands of individuals in a variety of urban public spaces. In general, telecommunications companies are now moving deliberately to monetize the location traces of their subscribers–it is thus quite likely that a crowd-tasking platform may obtain such network-based movement traces, especially if authorized by the individual crowd-worker.

Travel Itinerary Optimization: There has been significant research, in the decision optimization field, on spatiotemporal coordination of individual or group movement trajectories–this is known as the *orienteering problem* (Vansteenwegen, Souffriau, and Oudheusden 2011; Lin 2013; Vansteenwegen et al. 2009). A practical example of such a route-guidance based mobile App was recently presented in (Lau et al. 2012) for a theme park setting. The guidance algorithm solves a single-agent dynamic and stochastic variant of the orienteering problem, generating a *route* (i.e., sequence of attractions) which maximizes the user's rewards (tied to user preferences, wait times and travel times) subject to his start time and end time constraints and attraction wait time values. In our proposed *TRACCS* framework, instead of deriving a complete route, our goal is to devise a route (i.e., a sequence of location-centric tasks) that minimizes the additional detour from a predefined, daily lifestyle-based route. Moreover, we seek to derive optimal choices for thousands of such detour-minimizing routes jointly–creating, in effect, a sophisticated variant of the **team orienteering** problem (Chao, Golden, and Wasil 1996; Archetti, Hertz, and Speranza 2007; Chen, Cheng, and Lau 2013).

We do recognize that there is a large body of work focusing on solving pickup and delivery problems (e.g., see Baldacci, Bartolini, and Mingozzi (2011)) or dial-a-ride problems (e.g., see Cordeau and Laporte (2003)); both are similar to the mobile task assignment problem we intend to study in this paper. However, all existing models in these areas assume that a team of full-time staffs are employed and can carry out any specific order. Such assumptions are in direct conflict with the requirement that individual mobile workers should follow their respective predicted trajectories, and they could only spend limited amount of time deviating from their routes to perform assigned tasks. As such, none of existing model can be utilized straightforwardly.

The *TRACCS* Architecture & Assumptions

The *TRACCS* vision is centered around the use of *predicted location trajectories* of users for city-scale coordinated recommendation of tasks–this presupposes that the Crowd-Tasking Platform has access to the historical movement traces of individuals, so that it can create appropriate predictions for individual movement vectors. Figure 1 shows the overall architecture of the proposed *TRACCS* framework, and illustrates the various individual components and their interactions. The *Worker Interaction Manager* is responsible for handling interactions with individual workers via the mobile App (such as providing a list of suggested tasks, capturing user acceptance and completion of assigned tasks). The *Worker Profile Manager* is responsible for managing worker profiles (including tasks such as handling worker registration and indicating expertise for specific task categories). The *Task Manager* handles the interactions with task owners, allowing task owners to specify various attributes for tasks (such as the completion deadline, the amount of payment and the task location). The *Expected Route Predictor* utilizes historical traces of individual user movement to develop a predictive trajectory profile. Finally, the *Task Assignment Planner* is responsible for taking as inputs the list of location-dependent tasks, and the predicted movement profile of workers, and recommending an allocation of tasks (which individual workers may or may not accept) to each individual worker. To be upfront, this paper does **not focus on a real-world implementation**, but instead studies the *algorithmic techniques* needed by the *Task Assignment Planner* for scalable, good-quality coordinated task assignment.

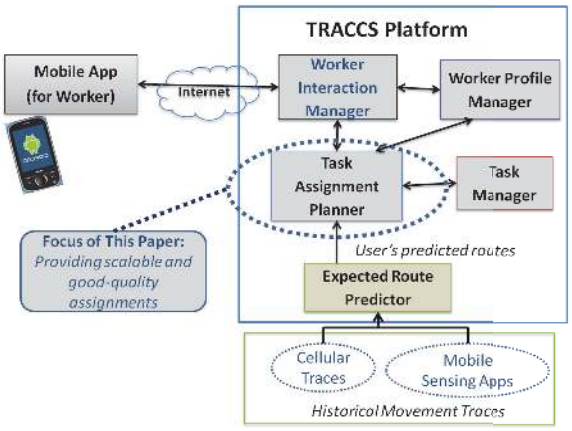

Figure 1: Overall architecture of the *TRACCS* framework.

Limiting Assumptions

Given our focus on establishing the feasibility of the coordinated allocation strategy, this paper makes the following assumptions to limit its scope:

- We assume that the Expected Route Predictor provides only a single routine trajectory (for the time interval specified–e.g., for each day) for each individual worker. In practice, each worker may have a small, but distinct set, of probabilistic alternatives (e.g., on leaving office, one can head for home, the gym or the club), resulting in a stochastic formulation of the optimization problem (deferred to future work).
- For the formulation considered here, we assume that each worker will accept and perform (and thus get rewarded for) *all* the tasks assigned to her, as long as the resulting detour overhead lies within her acceptable threshold. Clearly, significant additional work (including empirical studies) is needed to develop and incorporate "likelihood of task acceptance" models into the Task Assignment Planner.
- For our current work, we assume that tasks (and the associated rewards) are independent of one another. In certain scenarios, tasks may have certain group-based or sequential dependencies (e.g., both tasks T_3 and T_5 need to be completed before individual workers can be rewarded, or task T_3 must be completed prior to the initiation of task T_5). Such constraints on the feasible set of task assignments are not considered at present.

An Integer Linear Programming Model for the Mobile Crowdsourcing Problem

In the Introduction section, we mentioned two different modes of movement, detour-without-retrace and detour-with-rejoin, for planning mobile crowdsourcing task allocation. In this section, we propose an integer linear programming (ILP) model to capture the detour-with-rejoin model, and argue that with minor modifications, it can also be used for the detour-without-retrace model.

An ILP Model for Detour-with-Rejoin

We first define notations for model parameters:

- Let N be the set of all nodes, and M be the set of all mobile workers (agents). Let $t_{ij}, i, j \in N$ be the traveling time from node i to node j.
- Let $N_t \subset N$ be all task nodes. We assume that each task is represented by a unique task node. For each task $t \in N_t$, let s_t be the associated reward.
- Let $R^m \subset N$ be the set of agent m's routine nodes. For each node i in R^m, its intended visit sequence is denoted as p_i^m (e.g., if an agent's routine route starts from node 10, stops at node 3, and ends at node 7, we should set $p_{10}^m = 1, p_3^m = 2, p_7^m = 3$). Finally, each agent m should also specify the maximum detour time T^m.

The decision variables are:

- $x_{i,j}^m, i, j \in N, m \in M$, where $x_{i,j}^m = 1$ means that node i is traveled by agent m immediately before j.
- $z_i, i \in N$, where $z_i = 1$ means that node i is visited by at least one agent.
- $u_a^m = \{0, 1, \ldots, N\}, a \in \{1, 2, \ldots, 2(N-1)\}, m \in M$, denotes the node to be visited by agent m at order a. $u_a^m = 0$ is the special case where no node is visited at order a.

  The maximal number of nodes visited cannot exceed $2(N-1)$. The number $2(N-1)$ can be achieved if we have one routine node and $(N-1)$ task nodes, and we visit one task node and then come back to the routine node immediately before visiting the next task node.

The objective of our formulation is to maximize the total value obtained from visiting routine nodes (to ensure all agents stay on their respective courses) and task nodes:

$$\max \sum_{t \in N_t} s_t z_t. \tag{1}$$

The first group of constraints focus on enforcing routine routes and monitor the execution status of tasks, and are defined for all $m \in M$:

$$\sum_{i \in N} x_{n,i}^m \geq 1, n \in R_1^m, \tag{2}$$

$$\sum_{i \in N} x_{i,n}^m \geq 1, n \in R^m \setminus R_1^m, \tag{3}$$

$$\sum_{i \in N} x_{i,n}^m = \begin{cases} \sum_{i \in N} x_{n,i}^m - 1, n \in R_1^m, \\ \sum_{i \in N} x_{n,i}^m + 1, n \in R_2^m, \\ \sum_{i \in N} x_{n,i}^m, n \in N \setminus \{R_1^m \cup R_2^m\}, \end{cases} \tag{4}$$

$$z_t \leq \sum_{\hat{m} \in M} \sum_{i \in N} x_{i,t}^{\hat{m}}, t \in N_t, \tag{5}$$

$$\sum_{i,j \in N} t_{ij} x_{i,j}^m \leq T^m, \tag{6}$$

where agent m's origin and destination nodes are denoted respectively as two singleton sets $R_1^m = \{q | q \in R^m, p_q^m = 1\}$ and $R_2^m = \{q | q \in R^m, p_q^m = |R^m|\}$. With (2) and (3),

agent m is required to visit every routine node at least once. (4) ensures the flow conservation for origin, destination, and all other nodes. (5) sets the task execution flag z_n by aggregating all agent's traces. As z_n appears in the objective function, and this is a maximization problem, z_n will be set to 1 whenever possible.

Although the visit to routine nodes is guaranteed via the first group of constraints, the sequence of visit to routine nodes are not enforced. As such, the second group of constraints focus on constructing the order of traversal from the binary decision variable $x^m_{i,j}$, after which the desired sequence of visits to all routine nodes is then enforced. For the following set of constraints, we again assume that they are defined for all $m \in M$:

$$u^m_1 = n, n \in R^m_1, \tag{7}$$
$$(a+1) - b \leq K(1 - x^m_{u^m_a, u^m_b}), \forall a, b, \tag{8}$$
$$b - (a+1) \leq K(1 - x^m_{u^m_a, u^m_b}), \forall a, b, \tag{9}$$

where K is a large constant, and both a and b represent the order of visits. (7) ensures that the first node visited is always the origin node in agent m's routine route. The purpose of (8) and (9) is to ensure that $x^m_{i,j}$ cannot be set to 1 unless b is right after a (i.e., $b = a + 1$). This can be achieved since the only way to set right-hand sides to zero (i.e., set $x^m_{u^m_a, u^m_b}$ to be 1) while satisfying both (8) and (9) is to set left-hand side to zero as well (i.e., let $b = a + 1$).

Finally, we need to preserve the partial order defined on all routine nodes. In other words, if one node i is originally specified to be in front of another node j, the new order generated by our formulation should also preserve this order. In our formulation we achieve this by making sure that for any pair of routine nodes visited at orders a and b, the difference of a and b is at least as large as the difference in the original route:

$$a - b \geq p^m_{u^m_a} - p^m_{u^m_b}, \forall u^m_a, u^m_b \in R^m, a > b\ . \tag{10}$$

Note that (8)–(10) are all nonlinear constraints (since decision variables appeared as indices in problem data), and we have to linearize them so that the whole problem can be solved linearly.

Linearization To linearize (8)–(10), we need to define the following additional decision variables:

- $\beta^m_{i,l}$, where $\beta^m_{i,l} = 1$ if agent m visits node i at order l, and $\beta^m_{i,l} = 0$ otherwise.
- $\alpha^m_{i,j,l}$, where $\alpha^m_{i,j,l} = 1$ if agent m visits nodes i and j at orders l and $l + 1$ respectively. In other words, $\alpha^m_{i,j,l} = \beta^m_{i,l} \cdot \beta^m_{j,l+1}$.

(8) and (9) are first replaced by the linear constraints below (we again assume that the following constraints are defined for all $m \in M$):

$$x^m_{i,j} = \sum_{l=1}^{2N-3} \alpha^m_{i,j,l}, \forall i, j \in N, \tag{11}$$
$$\forall i, j \in N, l = 1, \ldots, 2N-3:$$
$$\begin{cases} \alpha^m_{i,j,l} \leq \beta^m_{i,l}, \\ \alpha^m_{i,j,l} \leq \beta^m_{j,l+1}, \\ \alpha^m_{i,j,l} \geq \beta^m_{i,l} + \beta^m_{j,l+1} - 1, \end{cases} \tag{12}$$
$$\sum_{i \in N} \beta^m_{i,l} \leq 1, l = 1, \ldots, 2(N-1), \tag{13}$$
$$u^m_l = \sum_{i \in N} i \cdot \beta^m_{i,l}, l = 1, \ldots, 2(N-1). \tag{14}$$

With (11)–(14), we can then proceed with the linearization of (10).

$$a - b \geq \sum_{i \in R^m} p^m_i (\beta^m_{i,a} - \beta^m_{i,b}), \tag{15}$$

where $b = 1, \ldots, 2N-3, a = b+1, \ldots, 2N-2, m \in M$.

The Detour-without-Retrace Variant

For the detour-without-retrace model, the ILP formulation is almost identical, except that we don't allow multiple visits to the same routine node (i.e., the '$\geq$' should be replaced by '$=$' in (2) and (3)). This significantly simplifies the ILP model, as each node can now be associated with a unique visit order. The implication of this is that we can replace decision variables u^m_a with v^m_i, which is the visit order of node i for agent m.

The collection of constraints (7)–(10) can thus be replaced by the following corresponding constraints ($m \in M$):

$$v^m_n = 1, n \in R^m_1, \tag{16}$$
$$(v^m_i + 1) - v^m_j \leq K(1 - x^m_{i,j}), i, j \in N, \tag{17}$$
$$v^m_j - (v^m_i + 1) \leq K(1 - x^m_{i,j}), i, j \in N, \tag{18}$$
$$v^m_i - v^m_j \geq p^m_i - p^m_j, i, j \in R^m, p^m_i > p^m_j. \tag{19}$$

Despite the similarity between (7)–(10) and (16)–(19), do note that (16)–(19) are already linear, thus no further linearization would be necessary. The resulting ILP formulation for the detour-without-retrace model is therefore much more compact and can be solved more efficiently. However, even with this much smaller and efficient formulation, solving the ILP formulation exactly becomes intractable even for numerical instances of moderate sizes (the largest instance we managed to solve with CPLEX consists of 15 agents and 100 nodes, and it took more than 7 hours to finish). As an exploratory approach, we implemented simple greedy and local search heuristics, and for small-scale problems, they performed very well both in terms of execution time and solution quality (for the same numerical instance above, our simple heuristic took less than a second to finish, while getting almost identical allocation result). As such, we have put most of our developmental efforts on designing efficient heuristics.

Extensions to Handle Additional Constraints

The basic ILP formulation above can be extended to capture several other variants of interest:

- *Multi-Worker Task Coverage*: In crowd-sourcing scenarios, it is commonplace to require that a particular task be performed by multiple workers, with task's final result being determined statistically (e.g., by majority voting) from all the assigned workers. If a task t requires τ_t workers to perform the task (i.e., visit the task node) before it can be considered completed, we can handle it by modifying (5) as:

$$z_t \leq \sum_{\hat{m} \in M} \sum_{i \in N} x_{i,t}^{\hat{m}} - \tau_t + 1, t \in N_t, \qquad (20)$$

  i.e., z_t (the flag indicating successful execution of task t) can only be set to 1 if $\sum_{\hat{m} \in M} \sum_{i \in N} x_{i,t}^{\hat{m}}$ is at least τ_t.

- *Prioritizing Tasks*: To handle situations where certain tasks are deemed more important than others, this ILP formulation can be modified to assign higher rewards s_t for higher priority tasks. While this does not provide strict priority guarantees, in practice, the reward maximizing formulation causes the worker detours to be biased in favor of tasks that offer higher rewards.

To focus on the key properties of our proposed framework, within the specified space constraints, we do not explore these relatively minor variants further in this paper. Instead, we present the heuristics we developed, as well as large-scale numerical experiments that are close to the kind of scale we would anticipate in real-world deployment.

Designing Efficient Heuristic for *TRACCS*

The ILP formulations we proposed for the detour-without-retrace model and the detour-with-rejoin model are shown to be intractable even for small problems. For our purpose of scheduling mobile crowd-sourcing tasks, we not only want to solve the problem well, we also expect to have stringent limits on execution time. In practice, most crowd-workers would want to see recommendations prepared for them within a few seconds after they turn on their App; similarly, task owners would like their tasks to be assigned shortly after they've been submitted to the crowd-tasking platform. To support such low-latency assignment, our heuristic design is divided into two phases: we first construct an initial solution as fast as possible by using a greedy heuristic; the quality of the initial solution is then improved iteratively by an iterated local search (ILS) when time permits.

Greedy Construction Heuristic

The greedy construction heuristic, outlined in Algorithm 1, is an intuitive heuristic for building an initial solution. As a start, all agents are assigned their full routine nodes following given orders. After that, a task node with the least travel time from any available agent (i.e., an agent whose detour threshold has not been exhausted) will be inserted into that agent's route. Task nodes without positive values will not be inserted. The above process is repeated until either all task nodes are assigned or no task can be feasibly inserted.

Algorithm 1 Greedy Construction Heuristic

```
procedure GREEDY(N_t, M, R^m, D)
    while N_t ≠Ø or no task t can be assigned do
        for each task t ⊂ N_t do
            cost*_t ← a very large number
            for each agent m in M do
                cost_mt ← COMPUTELEASTCOST(R^m, t)
                if remaining detour time for m ≥ cost_mt then
                    A ← ADDTOFEASIBLESET(m, t);
                    if cost_mt < cost*_t then
                        UPDATEFEASIBLELEASTCOSTROUTE(A);
                    end if
                end if
            end for
        end for
        {m, t} ← PICKFEASIBLELEASTCOSTROUTE(A);
        UPDATEROUTE(t, R^m);
    end while
    return solution set R^m
end procedure
```

Iterated Local Search (ILS)

ILS improves a complete solution generated by the greedy construction heuristic. This solution is initially treated as BESTFOUND. We consider four different operations, the swap, move, insert and replace operations (Table 1).

Notations	Descriptions
SWAP	Exchange two tasks between two routes.
MOVE	Move one task from one route to another route.
INSERT	Insert unassigned task into the routes.
REPLACE	Replace one assigned task with one unassigned the task.

Table 1: Notations and descriptions.

The first random improvement strategy is applied for the SWAP operation. Two agents with two task nodes are selected randomly. SWAP is executed if it increases the total remaining detour time for both agents. MOVE operation is performed by reallocating one randomly scheduled task from one agent to another agent with the highest remaining detour time. This task is then inserted to a node with the least time incurred. INSERT operation is started by choosing one unassigned task with the highest utility/reward score. We then select one agent with the highest remaining detour time and insert the unassigned task node with the least time incurred. REPLACE operation is used to replace one assigned task with one unassigned task. The operation is started by selecting one unassigned task with the highest utility score. An agent is then randomly selected. We examine all possible insertions by considering the feasibility and the highest ratio between OVERALLUTILITYINCREMENT/DETOURTIMEINCREMENT. The entire process is repeated until it exceeds MAXLOOP iterations. The solution would be updated if a better solution is obtained. Finally, the algorithm returns the best known solution obtained, BESTFOUND. The ILS can be seen in Algorithm 2.

Algorithm 2 Iterated Local Search Algorithm

```
procedure ILS(N_t, M, R^m, D)
    BESTFOUND ← GREEDY(N_t, M, R^m, D);
    LOOP = 0;
    while LOOP < MAXLOOP do
        SWAP(N_t, M, R^m, D);
        MOVE(N_t, M, R^m, D);
        INSERT(N_t, M, R^m, D);
        REPLACE(N_t, M, R^m, D);
        if SOLUTION better than BESTFOUND then
            Update BESTFOUND;
        end if
        Loop++;
    end while
    return BESTFOUND
end procedure
```

Experimental Results and Insights

In this section, we report a comprehensive suite of experimental results that help evaluate the *TRACCS* framework and our heuristic algorithms. The results evaluate performance metrics such as: the *Task Completion Ratio*, which indicates the fraction of tasks that have been successfully allocated to a crowd-worker (and thus implicitly serves as an indicator of the cumulative reward gained by all workers), the *Detour Overhead*, which measures the total time/distance overheads that the assigned tasks will impose on the workers, and the *Computation Time*, which measures the total execution time for each task assignment algorithm. To make our results meaningful, we first use a custom-built Instance Generator to create multiple instances of realistic urban trajectories and task locations, incorporating one or more suburban and central business locations. All experiments reported here were run on a 3.2 GHz Intel (R) Core (TM) Windows 7 machine, with 12GB of RAM. Unless otherwise mentioned, all results are computed as the average of 5 separate runs, each with distinct random seeds.

Instances Generated

The results in this study are based on two different types of topologies that share a fundamental commuting pattern (coming in from residential suburbs to a central business district in the morning, and performing the reverse commute in the evening), and differ principally in the number of distinct residential suburbs.

- *OneOrigin:* Here multiple workers travel from the same residential area to a single major business hub. The instances are generated with varying values of the following parameters: number of agents $|M|$ (individual crowd-workers), number of nodes $|N|$ (total number of distinct locations in the topology) and the task-to-worker ratio $|N_t|/|M|$ (where $|N_t|$ indicates the number of location-dependent tasks). Moreover, each worker has an associated *detour threshold* D (expressed as a percentage of the overall commuting distance), that indicates the maximum amount of task-related detour that the worker can tolerate. In general, D can be expressed either in distance or time; for our studies, we assume that the traveling speed is uniform, and thus express the detour threshold purely in terms of the additional travel distance. For example, a uniform value of $D = 0.1$ implies that each of the M workers can accept tasks as along as the total detour does not exceed 10% of the distance between its source and origin. The tasks are generated uniformly across the entire topology.
- *MultipleOrigins:* Here, the topology is generalized to incorporate multiple residential areas, with each worker traveling from her own individual residential area to reach the central business hub. Moreover, in this case, we also assume that (i) the workers pass through various intermediate points (corresponding to transfer points such as bus transfer stations or subway junctions), and that (ii) the tasks are not confined to just each residential area or the central business hub, but can be located at these intermediate transfer points as well. The distribution of tasks is expressed as a tuple of the form (p_r, p_i, p_h), where p_r, p_i and p_h represent the ratio of tasks in the residential area, intermediate points and the business hub, respectively. For the results reported in this paper, we use two standard scenarios: p_r = 50%,p_i = 30%, p_h = 20% and p_r = 60%,p_i = 20%, p_h = 20%, reflecting our expectation that a majority of the tasks will be associated with the central business hub.

Algorithms Compared

As we have already established that the accurate ILP-based formulation is computationally infeasible, except for very small instances (tens of workers and tasks), we compare the performance of three heuristics:

1. *Greedy:* The iterative algorithm that assigns each task to a worker who will experience the smallest possible additional detour, as a result of accepting this task.
2. *Greedy+ ILS:* The enhanced version of Greedy, that performs additional randomized changes in task assignments, to overcome local minima.
3. *Myopic:* In this *baseline* algorithm, the task assignment is performed sequentially, across workers, depending on their arrival time at specific locations. At any location, an arrival worker checks the list of all nearby available (unassigned tasks), and is assigned the maximum possible set of tasks that she can perform, without violating her detour threshold. Note that, in this approach, it is possible for some workers to corner the lion's share of the tasks, and also potentially to exhaust the total detour slack very early, even before reaching the destination area. The myopic baseline is designed to mimic the current practice of most mobile crowd-sourcing platform providers.

Task Completion and Detours Experienced

We first study the overall task completion rate and the average detour overhead experienced by the workers, for both the *OneOrigin* and *MultipleOrigins* models (both instances). For these studies, we varied the detour threshold D (assumed to be uniform for all workers) between 5-20% of each individual worker's regular commuting distance. Figures 2 and 3

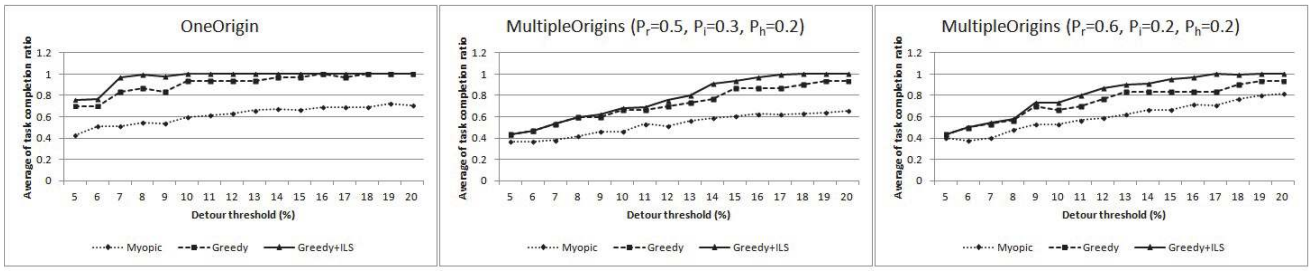

Figure 2: Plot of the task completion ratio.

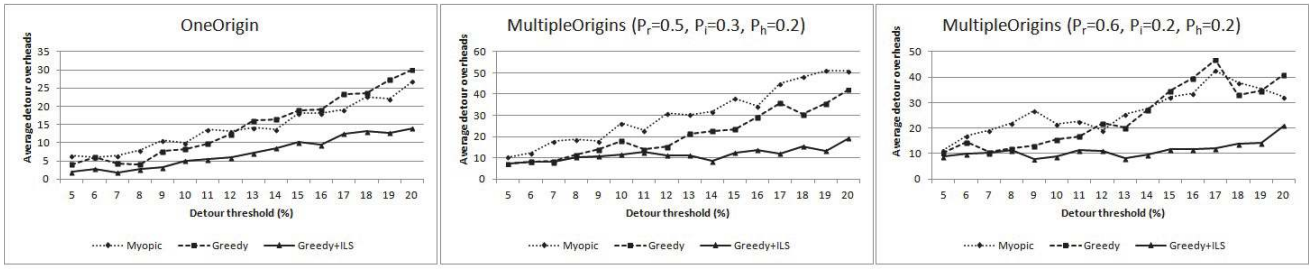

Figure 3: Plots of the average detour overheads.

show, respectively, the Task Completion Ratio and the Average detour overheads (averaged across all workers, and expressed as a percentage of the detour threshold) for all 3 heuristics, with the number of task workers $|M|$ equal to 10, the number of total nodes ($|N|$) equal to 100 and the number of task nodes ($|N_t|$) equal to 30.

While Figure 2 only shows the average values (and not the variation in the task completion ratios) across different instances, we additionally performed the paired t-Test (at 0.05 significance level) between each pair of algorithms, at every value of the detour threshold. We found the differences to be *statistically significant* (between *Greedy+ILS* and *Myopic*) for all values of D–i.e., *Greedy+ILS* alway clearly outperforms the *Myopic* strategy. Moreover *Greedy+ILS* is statistically better than simple *Greedy* in most cases, except for small values of D–if the detour threshold is too small, the set of detour opportunities is too limited for the additional randomized task swapping heuristics to make a significant difference.

From the figures, we can see that the *Greedy+ILS* scheme offers the best performance, assigning 100% of the tasks if the detour threshold is 10% or higher in the OneOrigin case and reaching close to 80% for thresholds of 10% and higher (for the *SingleOrigin* scenarios). The *Greedy+ILS* performs as much as 10-15% better, compared to the *Greedy* heuristic, with the performance gap increasing as D increases (i.e., as more flexibility in re-routing is permitted). More importantly, by looking at task assignment opportunities over a longer time-horizon (i.e., over the entire commuting itinerary), our heuristics outperform the typical *Myopic* alternative (often by 20% or higher), with this performance gap observable across both low and large values of D. This illustrates the biggest advantage of the *TRACCS* framework–*by better coordinating task assignment across multiple workers, we are able to complete an appreciably larger fraction of the total tasks, resulting in greater cumulative rewards for all the task workers*.

As expected, our heuristics also result in larger detour overheads as D increases–clearly, a larger value of D allows each worker to be assigned a larger set of assigned tasks, with corresponding longer detour overheads. However, note that the detour overheads for *Greedy+ILS* are significantly smaller than those for the *Myopic* approach, even though *Greedy+ILS* has a 20% or higher task assignment rate! *Clearly, the* TRACCS *framework is not only able to assign more tasks to more workers, but is also better at reducing the per-worker detour overhead.*

To illustrate the differences between the algorithms, Figure 4 plots the 3 trajectories (for two representative workers) computed on a specific topology instance, by each of the 3 heuristics.

Fairness Across Workers

To further investigate the finer details of task allocation offered by our heuristics, we carefully studied the detour overheads experienced by each worker. As an illustrative example of our result, Figure 5 plots the box-plots of the total detour overhead (as a percentage of the total detour threshold) for *each worker*, for the 3 different assignment strategies, for three distinct values of D (5%, 10% and 15%, respectively). It is clear that *Greedy+ILS* has the lowest *variation* among different workers, while *Myopic* results in the largest inter-worker variance. Clearly, our assignment algorithms not only offer higher cumulative rewards, but also provide a *more equitable* sharing of the detour burden among the workers. This may mitigate the poor mass adoption of crowd-sourcing pointed out by Musthag and Ganesan (2013), and make mobile crowd-tasking more widely

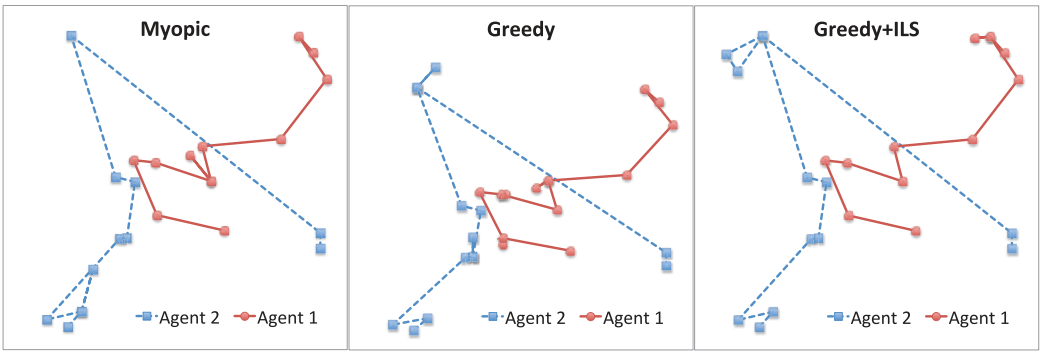

Figure 4: Routine trajectory & detour illustrated.

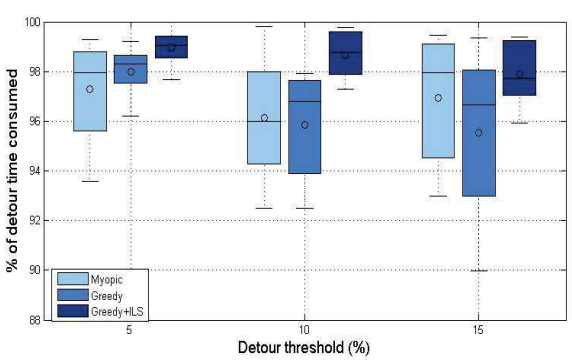

Figure 5: Inter-worker variance in detour overheads.

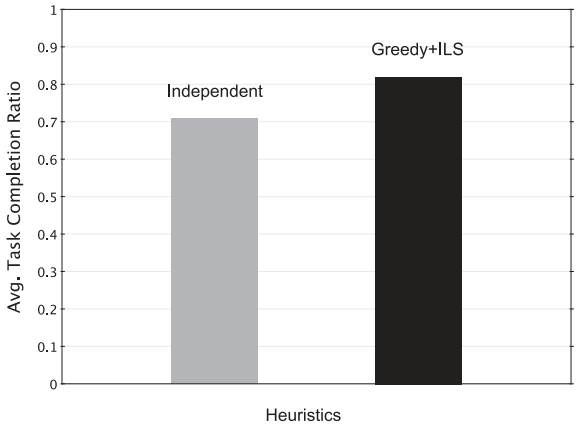

Figure 6: Independent and *Greedy+ILS*.

adopted by a larger share of urban workers.

The Need for Global Coordination Our proposed *TRACCS* framework assigns tasks to multiple workers in a coordinated fashion. This should result not just in lower unfairness among workers (as shown above) but also better overall rates of task completion. To isolate the impact of such global vs. independent coordination, we also implemented an additional, more-sophisticated, decentralized algorithm called *Independent*.

In this *Independent* approach, we perform task allocation sequentially (i.e., one worker at a time), but unlike *Myopic*, each worker assigns itself the best set of tasks while taking into account her *entire trajectory*. More specifically, one worker will be randomly drawn from the work force and she will be assigned the maximum possible available set of tasks (i.e., more specifically, the tasks that maximize her cumulative reward) that can be performed, without violating the detour threshold. This process continues iteratively (one worker at a time), with successive workers maximizing their rewards from only the set of currently un-allocated tasks.

By incorporating the notion of limited detours from the routine trajectory, *Independent* helps us isolate the benefit of performing task allocation in a coordinated fashion. We studied its performance versus *Greedy+ILS* and discovered that, in many topologies, *Independent* approach can lead to very sub-optimal task allocations, where some workers preferentially pick the set of tasks that could have been performed by other workers, thereby leaving these other workers with a residual set of mostly infeasible tasks. Figure 6 illustrates one such scenario, for a topology with (p_r, p_i, p_h) = (0.1, 0.7, 0.2)–i.e., where most tasks are distributed in the intermediate commuting regions. We see that the task completion ratio of *Independent* (around 0.7) is significantly lower than that achieved via the centralized *Greedy+ILS* approach (around 0.8).

Scalable Growth in Complexity

To demonstrate the ability of our heuristics to scale to large numbers of either crowd-workers or location-sensitive tasks, we performed extensive experimental studies, with various values of $|M|$ and the task-to-worker ratio ($|N_t|/|M|$). Table 2 summarizes the total assignment computation time, and the task completion ratio, taken by the *Greedy+ILP* heuristic, for different values of $|M|$ and the task completion ratio, for a representative topology instance: *MultipleOrigins(0.5,0.3,0.2)*, with $D = 10\%$.

We can see that the assignment algorithm takes no more than 7.73 seconds to compute for even relatively large sizes (1000 workers and 3000 tasks). Moreover, we see that the growth in computation time is *sub-linear* in the number of workers $|M|$ (a 50-fold increase in $|M|$ from 20 to 1000 results in only a 5-fold increase in the computation time,

from 0.4 to 2.0 secs). Moreover, in each case, we see that the *Greedy+ILS* is able to achieve task completion rates of 0.68 or higher, and maintains its high task completion rates even as the size of the problem increases. While real-world studies will definitely be more conclusive, our results lead us to believe that our heuristics can handle urban-scale crowd-tasking problems (involving, several thousands of crowd-workers and location-sensitive tasks).

Handling Worker Diversity

To demonstrate that our heuristics can handle heterogeneous worker populations, we also ran experiments with multiple worker classes, characterized by their respective detour threshold limits D. To measure the impact of D on task assignments to different worker classes, we define *per-worker task assignment ratio* for each worker, which is computed as the ratio of the number of assigned tasks over the fair assignment (the fair assignment is number of tasks per worker, i.e., $|N_t|/|M|$). By computing the averages of per-worker task assignment ratio for each worker classes, we can quickly estimate the relative dominance of a worker class in receiving task assignment.

As an illustrative example, we use a multi-origin network with parameters $(p_r = 0.6, p_i = 0.2, p_h = 0.2)$, and let $|N_t| = 30$ tasks and $|M| = 10$ workers, where we have 2 classes of 5 workers each, one with $D = 10\%$ and another with $D = 20\%$. Based on earlier discussion, the fair assignment in this case should be 3. The average per-worker task assignment ratios for all three sets of heuristics are plotted in Figure 7. For Myopic, Greedy, and Greedy+ILS heuristics, numbers of assigned tasks are 20, 24, and 28 respectively. And in all cases, as expected, the worker class with higher detour limit ($D = 20\%$) has higher ratios than the other worker class (with $D = 10\%$), and the advantages of 20% class over 10% class are 50%, 67%, and 111% respectively.

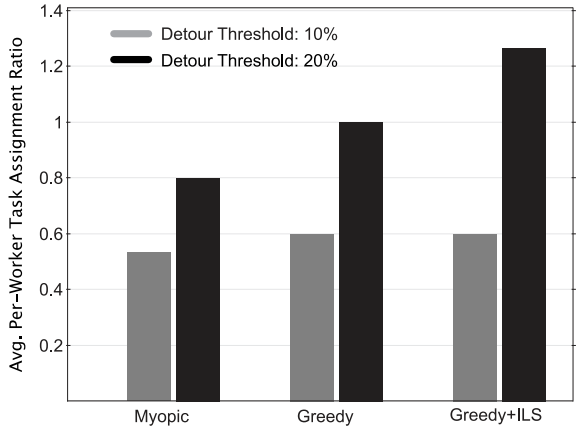

Figure 7: *Greedy+ILS* with two heterogeneous worker classes: $D = 10\%$ and $D = 20\%$. The network instance is multi-origin with parameters: $(p_r = 0.6, p_i = 0.2, p_h = 0.2)$. $|N_t| = 30$ tasks and $|M| = 10$ workers.

We can see that the Greedy+ILS heuristic is again capable of finding better solution, and it achieves so by better utilizing detour limits offered by different worker classes.

Conclusions

We have presented *TRACCS*, a new *centrally-coordinated* approach for assigning mobile crowd-sourced tasks in urban environments. We formulated the assignment problem as one of large-scale optimization, which seeks to maximize the cumulative rewards for all assigned tasks, while ensuring that each individual's task-related *detour* (from their normal movement trajectory) stays within a specified bound. Having established the computational difficulty of the exact spatiotemporal optimization problem, we then presented two heuristic approaches for scalable task assignment. Detailed experimental studies show that, for our synthetically-generated, but realistic, topologies and movement patterns: (i) our proposed *Greedy+ILS* heuristic is able to typically assign over 85-90% of the total tasks when workers are willing to tolerate detours that are no higher than 10% of their normal commuting time or distance, and, more importantly, (ii) our centrally-coordinated heuristics significantly outperform the present *Myopic* approach of independent, opportunistic selection by individual crowd-workers, achieving not just 20% or higher task assignment rates, but providing fairer task allocation across all workers and reducing the average detour overhead per worker by 60% or higher.

While these results are quite promising, there are significant opportunities for further improvement and refinement of the *TRACCS* framework. To make the task assignment algorithms applicable to a wider variety of problem settings, the algorithms have to be enhanced to deal with not just additional constraints (such as sequential dependencies among tasks, or finer-grained time separation requirements for recurring tasks), but to also incorporate the statistical forecasting uncertainty about the specific trajectory of each worker. To empirically establish the performance gains with this approach, we are presently working to build and deploy an experimental mobile crowd-tasking App (containing implementations of our task-assignment heuristics) to a pool of over 1000 participants in a real-world urban setting. Only such real-world deployments can reveal whether task-workers will adopt this paradigm, and how their task acceptance and completion rates will be affected by our strategy of providing them a *sequence of tasks* ahead of time.

Acknowledgment

This material is based on research sponsored in part by the Air Force Research Laboratory, under agreement number FA2386-14-1-0002. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors.

References

Archetti, C.; Hertz, A.; and Speranza, M. G. 2007. Metaheuristics for the team orienteering problem. *Journal of Heuristics* 13(1):49–76.

agents	nodes	task ratio=2		task ratio=3	
		Computation time (in seconds)	Task completion ratio	Computation time (in seconds)	Task completion ratio
10	100	0.367	0.75	0.322	0.68
20	200	0.382	0.885	0.382	0.75
250	1250	0.952	0.997	1.626	0.914
500	2500	0.974	1	3.29	0.947
1000	5000	2.029	0.986	7.72	0.857

Table 2: Scalability of the *Greedy-ILS* heuristic.

Baldacci, R.; Bartolini, E.; and Mingozzi, A. 2011. An Exact Algorithm for the Pickup and Delivery Problem with Time Windows. *Operations Research* 59(2):414–426.

Becker, R.; Cáceres, R.; Hanson, K.; Isaacman, S.; Loh, J. M.; Martonosi, M.; Rowland, J.; Urbanek, S.; Varshavsky, A.; and Volinsky, C. 2013. Human mobility characterization from cellular network data. *Communications of the ACM* 56(1):74–82.

Chao, I.-M.; Golden, B. L.; and Wasil, E. A. 1996. The team orienteering problem. *European Journal of Operational Research* 88(3):464–474.

Chen, C.; Cheng, S.-F.; and Lau, H. C. 2013. The multi-agent orienteering problem. In *Tenth Metaheuristics International Conference*.

Cordeau, J.-F. o., and Laporte, G. 2003. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* 1(2):89–101.

FieldAgent. http://www.fieldagent.net/. Accessed April 2014.

GigWalk. http://gigwalk.com/. Accessed April 2014.

Kazemi, L., and Shahabi, C. 2012. Geocrowd: enabling query answering with spatial crowdsourcing. In *Twentieth International Conference on Advances in Geographic Information Systems*, 189–198. ACM.

Kokkalis, N.; Köhn, T.; Huebner, J.; Lee, M.; Schulze, F.; and Klemmer, S. R. 2013. Taskgenies: Automatically providing action plans helps people complete tasks. *ACM Transactions on Computer-Human Interaction* 20(5):27.

LaMarca, A.; Chawathe, Y.; Consolvo, S.; Hightower, J.; Smith, I.; Scott, J.; Sohn, T.; Howard, J.; Hughes, J.; Potter, F.; Tabert, J.; Powledge, P.; Borriello, G.; and Schilit, B. 2005. Place lab: Device positioning using radio beacons in the wild. In *Third International Conference on Pervasive Computing*, PERVASIVE'05.

Lau, H. C.; Yeoh, W.; Varakantham, P.; Nguyen, D. T.; and Chen, H. 2012. Dynamic stochastic orienteering problems for risk-aware applications. In *Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 448–458.

Lin, S.-W. 2013. Solving the team orienteering problem using effective multi-start simulated annealing. *Applied Soft Computing* 13(2):1064–1073.

mCrowd. https://crowd.cs.umass.edu/. Accessed April 2014.

Misra, A., and Balan, R. K. 2013. Livelabs: Initial reflections on building a large-scale mobile behavioral experimentation testbed. *SIGMOBILE Mobile Computing and Communications Review* 17(4):47–59.

Musthag, M., and Ganesan, D. 2013. Labor dynamics in a mobile micro-task market. In *SIGCHI Conference on Human Factors in Computing Systems*, 641–650.

NeighborFavor. http://www.crunchbase.com/company/neighbfav. Accessed April 2014.

Talamadupula, K.; Kambhampati, S.; Hu, Y.; Nguyen, T. A.; and Zhuo, H. H. 2013. Herding the crowd: Automated planning for crowdsourced planning. In *First AAAI Conference on Human Computation and Crowdsourcing*, 70–71.

Vansteenwegen, P.; Souffriau, W.; Vanden Berghe, G.; and Van Oudheusden, D. 2009. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* 36(12):3281–3290.

Vansteenwegen, P.; Souffriau, W.; and Oudheusden, D. V. 2011. The orienteering problem: A survey. *European Journal of Operational Research* 209(1):1–10.