Towards City-scale Mobile Crowdsourcing: Task Recommendations under Trajectory Uncertainties*

Cen Chen **Shih-Fen Cheng** **Hoong Chuin Lau** **Archan Misra**
School of Information Systems, Singapore Management University
80 Stamford Road, Singapore 178902
{cenchen.2012, sfcheng, hclau, archanm}@smu.edu.sg

Abstract

In this work, we investigate the problem of large-scale mobile crowdsourcing, where workers are financially motivated to perform location-based tasks physically. Unlike current industry practice that relies on workers to manually pick tasks to perform, we automatically make task recommendation based on workers' historical trajectories and desired time budgets. The challenge of predicting workers' trajectories is that it is faced with uncertainties, as a work does not take same routes every day. The challenge of predicting workers' trajectories is that it is faced with uncertainties, as a worker does not take same routes every day. In this work, we depart from deterministic modeling and study the stochastic task recommendation problem where each worker is associated with several predicted routine routes with probabilities. We formulate this problem as a stochastic integer linear program whose goal is to maximize the expected total utility achieved by all workers. We further exploit the separable structures of the formulation and apply the Lagrangian relaxation technique to scale up computation. Experiments have been performed over the instances generated using the real Singapore transportation network. The results show that we can find significantly better solutions than the deterministic formulation, and in most cases we can find solutions that are very close to the theoretical performance limit.

1 Introduction

In the traditional crowdsourcing paradigm (introduced by Howe and Robinson in 2005 and elaborated in [Howe, 2006]), task owners distribute tasks (or sub-parts, often referred to as"microtasks") via online platforms to attract crowdworkers who would work on them for small payoffs. With the proliferation of platforms such as Amazon Mechanical Turk and CloudFactory, this paradigm has become an important and integral part of many business operations and processes, as well as led spawned the emergent field of *human computation*, where AI researchers study issues related to coordination, incentive design, process design, task assignment optimization and even ethics.

Mobile crowdsourcing is a rapidly growing extension to the traditional crowdsourcing paradigm, characterized by (a) tasks having strong *location specificity* (i.e., the task requires a crowdworker physically visiting a specific location) and (b) tasks principally requiring the use of mobile devices/smartphones at these location. Examples of mobile crowdsourcing tasks include citizen sensing (ask participants to contribute sensor readings such as pollution, congestion, noise level) [Kanhere, 2011], price checks, store audits (e.g., checking shelves, store displays), logistics (package pickup and delivery), crowd estimates (at movie theaters and food courts), to name a few.

Most academic research and practical deployments of mobile crowdsourcing presently employ a *pull-based* model, such that individual workers independently search through, and select from, the corpus of available tasks (often with built-in proximity filters that enable them to identify jobs that are close to their current location).As pointed out by [Musthag and Ganesan, 2013], such pull-based embodiments of mobile crowdsourcing, as in all other forms of crowdsourcing, suffer from the phenomenon of *super agents*; i.e., a small percentage of crowdworkers who perform the majority of tasks[1]. Domination at such high level is undesirable, as many ordinary crowdworkers might drop out as a result of not having enough tasks, reducing the worker pool and thus the peak capacity of the mobile crowdsourcing platform. By examining empirical data, Musthag and Ganesan conclude that the major difference between ordinary worker agents and super agents is the latter's ability in planning better routes and choosing tasks that fit their routes best. A recent work by [Chen *et al.*, 2014] builds upon this insight and investigates an alternative *push-based* mobile crowdsourcing model, where the crowdsourcing platform can centrally plan for task recommendation, taking into account each mobile crowdworker's likeliest daily *routine route*. Their proposed

*This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority(MDA), and partially supported by the Xerox Research Centre India.

[1]Based on their study, 10% of most active users complete 80% of all completed tasks.

system, TRACCS, can not only enable more tasks to be completed with less time spent in *detours*, recommended tasks are also more evenly distributed, thus reducing the negative impacts of super agents. However, they make the simplifying assumption that each crowdworker has one and only one *deterministically-known* routine route.

In this paper, we extend this recently proposed work on *push-based* mobile crowdsourcing, to incorporate more realistic scenarios where an individual worker's movement trajectory has inherent uncertainty (as the worker may travel on different routes on different days). Such uncertainty presents a technical challenge to effective task recommendation under the push-based model, as the recommendation strategy must still seek to recommend individual workers (modeled as independent agents) tasks that are likely to lie along their routine commuting routes, while accounting for and trying to mitigate the likelihood that their actual routes (on a specific day) might make it infeasible for them to perform one or more of the recommended tasks.

Our contributions to this novel problem of *multi-agent task recommendation, under stochastic spatiotemporal uncertainty* are twofold:

- First, we show that this task recommendation problem can be formulated as a stochastic integer linear programming model, assuming that each agent's list of possible routine trajectories is finite, and governed by a known probabilistic distribution.
- Second, we develop and evaluate a combination of heuristics to address this computationally intractable problem. More specifically, we exploit the problem's structure to decompose the problem into independent agent-specific routing subproblems (each of which can be solved very efficiently), where the global coupling constraint is relaxed using the Lagrangian relaxation framework. Applying this computationally tractable framework to representative city-scale topologies, we show that we can not only obtain significant improvement over the deterministic model (around 8% – 15%), but also that the performance is almost identical to the cases where perfect information is available (i.e., where each agent's routine route realization is assumed to be exactly known).

2 Related Work

Mobile crowdsourcing: Recent approaches such as [Alt *et al.*, 2010], [Kazemi and Shahabi, 2012], and [Kazemi and Shahabi, 2011] have focused on a particular class of mobile crowdsourcing, called participatory sensing, with a goal of maximizing the number of assigned tasks based on agents' current locations. For example, [Kazemi and Shahabi, 2012] developed a centralized allocation algorithm focused on maximizing an agent's set of allocated tasks, while satisfying a proximity constraint (which implied that some tasks could be potentially performed by multiple agents). [Sadilek *et al.*, 2013] studied a general class of problems called crowdphysics where tasks requires people to collaborate in a synchronized space and time. They reduced the problem into a graph planning problem and proposed two methods: global coordination using shortest-path algorithm and opportunistic routing based on the ranking of the time-stamped locations.

Researchers have also investigated the impact of incentives on task assignment/selection behavior in mobile crowdsourcing. [Rula *et al.*, 2014] investigated the relative impacts of two different incentive mechanisms—micro-payments vs. weighted lotteries, on the task acceptance and completion rates of mobile agents (crowd-workers). The MSensing approach by [Yang *et al.*, 2012] focuses on incentive design for mobile crowdsourcing tasks, employing either a Stackelberg game model (for scenarios where the reward is specified by the platform) or an auction-based agent selection mechanism (where the reward is determined through competitive bidding) to model the dynamics between the pricing of individual tasks and the willingness of each agent (task worker) to perform that task. However, this approach either considers a single task in isolation or employs a cost function (for each agent) that fails to account for the regular movement trajectory of each agent. Other work on task assignment in mobile crowdsourcing addresses questions of reliability – for example, recently [Boutsis and Kalogeraki, 2014] seek to maximize the reliability of crowdsourcing tasks by selecting agents, subject to a task completion latency constraint.

In all of these approaches, the feasibility of task assignment is defined by the task's proximity to the agent's current location. In contrast, our focus is on a coordinated mechanism for *task recommendation* that operates over a longer time horizon (e.g., a day). We also explicitly model the inherent stochasticity (uncertainty) in individual agent trajectories and seek to minimize their task-related detours.

Orienteering problem: The planning of an agent's route to perform tasks that maximizes utility can be seen as the Orienteering Problem (OP) or Prize-Collecting Traveling Salesman Problem. The goal is to find a route that maximizes utility while respecting budget constraints. This problem was originally defined in [Tsiligirides, 1984] and motivated by the scheduling of a cross-country sport in which participants get rewards from visiting a predefined set of checkpoints. Since then, OP has been studied extensively by the Operations Research and AI communities and recently, [Vansteenwegen *et al.*, 2011] provided a survey of the formulations and solution approaches for the OP and its variants.

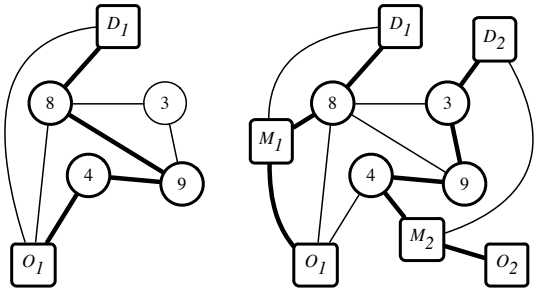
The closest variant of OP to our problem is the team orienteering problem (TOP), where a team of agents are scheduled to visit different locations to collect reward within a time budget [Chao *et al.*, 1996]. However, to the best of our knowledge, none of the previous research to date on TOP actually considers the incorporation of agent-specific location information, namely the probability distribution of routine routes taken by agents. One other variant of OP is the Multi-Period Orienteering Problem with Multiple Time Windows (MuPOPTW, studied by [Tricoire and Hartl., 2010]). In MuPOPTW, sales representatives need to visit a list of mandatory customers on a regular basis, while non-mandatory customers located nearby should also be considered and integrated into the current customer tours. While one may view the set of mandatory customers as the nodes on the routine routes and non-mandatory customers as the tasks in our problem, there is no predefined visiting sequence for

the mandatory customers and the stochasticity of agent routine routes is not captured in that model.

3 The Model

As mentioned in Section 1, we are interested in creating models for recommending tasks to mobile crowdworkers with known routine route distributions, so as to maximize the expected total rewards collected by all agents. The major constraints in task recommendation for each individual agent are the maximal amount of detour time allowed, the routine route that specifies the list of nodes each agent needs to traverse in order, and the probability distribution over a collection of routine routes (if this agent has multiple routine routes).

Mathematically speaking, this can be viewed as a specialized routing problem with time budget constraint and routine route requirement, and can be modeled as a variant of the well-known *orienteering problem* (OP). The classical OP can be seen in Figure 1a. In the classical OP, usually the requirement is for an agent to begin his trip from a given origin O_1, and to end at a given destination D_1. An agent can visit any node in-between O_1 and D_1 and incur corresponding link travel costs; yet he cannot exhaust his time budget before reaching D_1. The objective is for an individual agent to maximize value collected at visited nodes (each node comes with different value). The OP variant for the mobile crowdsourcing problem with deterministic routine route (for easy explanation) can be seen in Figure 1b. There are two major differences when compared to the classical OP in Figure 1a: 1) there are multiple agents involved, and the planner aims to optimize the sum of all agent's values; and 2) for each agent, a routine route is specified, and its partial order needs to be followed (e.g., in Figure 1b, agent 1 needs to follow $O_1 - M_1 - D_1$, while agent 2 needs to follow $O_2 - M_2 - D_2$). In both cases, all nodes can be visited at most once.



(a) Classical OP. (b) OP variant for mobile crowdsourcing.

Figure 1: Illustrations of OP variants.

To model uncertainty on agent's routine route, we assume that each agent may take one of multiple routes with known probability distribution. The objective is to optimize the sum of each agent's *expected* collected values. For the rest of the section, we will introduce an integer linear programming (ILP) formulation for the problem of mobile crowdsourcing with routine route uncertainty.

Note that our solution approach generates task recommendations for agents prior to the realization of actual routes taken. Such approach requires no agent inputs, and thus can reduce agent's cognitive burden in picking tasks; such approach is thus superior to the alternative where an agent has to first specify a deterministic route, before receiving task recommendation (this was the model in [Chen *et al.*, 2014]).

3.1 Notations

Let N be the set containing both the routine nodes and the task nodes (denoted as N_t), and for all pairs (i, j), where $i, j \in N$ and $i \neq j$, let t_{ij} be the corresponding travel time to move from i to j. Let K be the set of agents, and let M_k be the set of agent k's routine routes. For each route $m \in M_k$, let β_k^m be the probability that agent k would use route m, R_k^m be the collection of all nodes in route m, o_k^m be the origin, d_k^m be the destination, and p_{ik}^m be the visit order for node $i \in R_k^m$. For each task $i \in N_t$, let s_i be its reward, and e_i be its required execution time. The upper bound of the total detour time along route m for agent k is b_k^m.

We have the following four sets of decision variables:

- $y_{ik} \in \{0, 1\}$: set to 1 when task i is recommended to agent k.
- $x_{ijk}^m \in \{0, 1\}$: set to 1 when agent k moves from nodes i to j when the realized routine route is m.
- $u_{ik}^m \in \{0, \ldots, N\}$: indicates the visit order of node i for agent k, when the realized routine route is m.
- $z_{ik}^m \in \{0, 1\}$: set to 1 when agent k fails to complete recommended task i, when the realized routine route is m; if task i is not recommended to agent k in the first place, z_{ik}^m is set to 0 for all $m \in M_k$.

Note that the first set of decision variables, y_{ik}, is planner's recommendation decision; while the rest of decision variables are used for evaluating the outcome of the recommendation under different routine route realizations for each agent.

3.2 The Integer Programming Formulation

The objective of the integer programming formulation is to maximize expected total rewards earned by all agents, considering uncertainties over their routine routes. The quantity $y_{ik}(1 - \sum_{m \in M_k} \beta_k^m \cdot z_{ik}^m)$ refers to the probability that a task i can be finished, if it is recommended to agent k (i.e., if y_{ik} is set to 1). Hence, the objective is to:

$$\max \sum_{i \in N_t} s_i \sum_{k \in K} y_{ik} \left(1 - \sum_{m \in M_k} \beta_k^m \cdot z_{ik}^m\right). \qquad (1)$$

The above objective function is not linear, and in the next subsection, we will propose a linearized model.

Constraint (2) ensures that each task is recommended to at most one agent.

$$\sum_{k \in K} y_{ik} \leqslant 1, \qquad \forall i \in N_t. \qquad (2)$$

All other constraints are at agent-route level (k, m), i.e., for each agent, $k \in K$, and for each of his routine route realization, $m \in M_k$, the same set of constraints applies. These constraints are explained as follows.

The first group of constraints ensures that flows are consistent at all nodes. In particular, (3) specifies that inflow and outflow at any node d must be balanced (except for the origin and destination nodes) . (4) specifies that all routine nodes must be visited exactly once, while all other nodes can be visited by at most once.

$$\sum_{i \in N} x^m_{idk} = \sum_{j \in N} x^m_{djk}, \qquad \forall d \in N \backslash \{o^m_k, d^m_k\}, \quad (3)$$

$$\begin{cases} \sum_{j \in N} x^m_{djk} \leqslant 1, & \forall d \in N \backslash R^m_k, \\ \sum_{j \in N} x^m_{djk} = 1, & \forall d \in R^m_k \backslash \{d^m_k\}, \\ \sum_{i \in N} x^m_{idk} = 1, & d = d^m_k. \end{cases} \quad (4)$$

The time budget constraint for each routine route is enforced in (5).

$$\sum_{i \in N} \sum_{j \in N} (t_{ij} + e_i) \cdot x^m_{ijk} \leqslant b^m_k. \quad (5)$$

The following group of constraints produces visit orders (u^m_{ik}) from flows (x^m_{ijk}), and ensures that all nodes in the routine route m are visited according to the given sequence p^m_{nk}. In particular, (6) states that the visit order of the origin node must be 1; (7) states that if j is visited immediately after i (i.e., $x^m_{ijk} = 1$), the visit order of j should be *at least* 1 more than the visit order of i (i.e., $u^m_{jk} \geq (u^m_{ik} + 1)$).

$$u^m_{ik} = 1, \qquad i = o^m_k, \quad (6)$$

$$(u^m_{ik} + 1) - u^m_{jk} \leqslant N \cdot (1 - x^m_{ijk}), \qquad \forall i, j \in N, \quad (7)$$

(8) states that the partial order between any pair of nodes in the routine route must be preserved.

$$u^m_{ik} - u^m_{jk} \geqslant p^m_{ik} - p^m_{jk}, \quad \forall i, j \in R^m_k \ \& \ p^m_{ik} > p^m_{jk}. \quad (8)$$

Finally, (9) extracts whether a task node is bypassed ($z^m_{ik} = 1$) from the flow decision (x^m_{ijk}).

$$z^m_{ik} \geqslant 1 - \sum_{j \in N} x^m_{ijk}, \qquad \forall i \in N_t. \quad (9)$$

3.3 Linearization

As noted earlier, the objective function (1) is nonlinear, as it includes multiplicative terms composed of y_{ik} and z^m_{ik}, both of which are decision variables. We would linearize (1) by introducing δ^m_{ik} to replace z^m_{ik}:

- δ^m_{ik} is set to 1 if task i is recommended to agent k, yet cannot be completed when the realized routine route is m, and 0 otherwise. In other words, $\delta^m_{ik} = y_{ik} \cdot z^m_{ik}$.

With δ^m_{ik}, we can rewrite (1) as:

$$\max \sum_{i \in N_t} s_i \sum_{k \in K} \left(y_{ik} - \sum_{m \in M_k} \beta^m_k \cdot \delta^m_{ik} \right). \quad (10)$$

To characterize δ^m_{ik}, we would rewrite (9) for all (k, m) as:

$$\delta^m_{ik} \geqslant y_{ik} - \sum_{j \in N} x^m_{ijk}, \qquad \forall i \in N_t. \quad (11)$$

With the above modifications, the formulation is now linear, and can be solved as an ILP using standard commercial solvers such as CPLEX.

3.4 Scalability of the Model

The above ILP model can be solved exactly using CPLEX. However, it cannot scale to realistic sizes (we are looking at city-scale deployment). We nonetheless introduce this exact model for two purposes: 1) to enable the design of heuristic that exploits the problem structure revealed in the ILP model (to be described in Section 4), and 2) to provide a benchmark for the designed heuristic.

To make the ILP model more scalable (it would benefit both objectives listed above), we propose the following performance-boosting preprocessing procedures on the data without affecting the optimality of the model.

- For each subproblem pair (k, m), we would remove all tasks that cannot be reached within the given detour time. Therefore, instead of using global node sets N and N_t in (k, m)-level constraints, we would use (k, m)-specific node sets N^m_k and N^m_{tk} instead.
- To avoid having to solve shortest path routing explicitly in the ILP model, we pre-compute all-pair shortest path distance, and refer to it as a N-by-N distance matrix. With this matrix, we can further eliminate all non-essential nodes from N^m_k. More specifically, N^m_k only needs to contain nodes along the routine route m and the feasible task nodes N^m_{tk}.

With these preprocessing steps, we are now ready to formally introduce the Lagrangian relaxation heuristic for the ILP.

4 Lagrangian Relaxation

In this section, we present a LR-based heuristic for our problem, and show that it is computationally efficient in producing high quality solutions.

In our ILP formulation, (11) is the set of complicating constraints that couples all agent-route subproblems together; therefore, this is the set of constraints we choose to dualize. Following the convention in the standard LR literature, we define $\boldsymbol{\lambda} = \{\ldots, \lambda^m_{ik}, \ldots\}$ to be the vector of Lagrangian multipliers associated with each and every constraint in (11) (each constraint is indexed as (k, m, i)), and convert the objective function to be a minimization function.

Denote the Lagrangian dual problem as $L(\boldsymbol{\lambda})$, with the following objective function:

$$\min \sum_{i \in N_t} s_i \sum_{k \in K} \left(\sum_{m \in M_k} \beta^m_k \cdot \delta^m_{ik} - y_{ik} \right) + \sum_{i \in N_t} \sum_{k \in K} \sum_{m \in M_k} \lambda^m_{ik} \left(y_{ik} - \delta^m_{ik} - \sum_{j \in N} x^m_{ijk} \right). \quad (12)$$

All constraints except (11) remain the same. However, by observing the problem structure, we can further decompose $L(\boldsymbol{\lambda})$ into two different classes of subproblems. The first is the *assignment subproblem*, which decides how tasks should be recommended to individual agents (focus on y_{ik}); the second is the *routing subproblem*, which finds the exact node

visit sequence for each (k, m) tuple (focus on x_{ijk}^m). The assignment subproblem is defined as:

$$\min \sum_{i \in N_t} s_i \sum_{k \in K} \left(\sum_{m \in M_k} \beta_k^m \cdot \delta_{ik}^m - y_{ik} \right) + \sum_{i \in N_t} \sum_{k \in K} \sum_{m \in M_k} \lambda_{ik}^m \left(y_{ik} - \delta_{ik}^m \right), \quad (13)$$

$$\delta_{ik}^m \leqslant y_{ik}, \qquad \forall i \in N_t, m \in M, k \in K, \quad (14)$$

together with the constraint set (2). (14) is included to further tighten this subproblem such that incompletion penalty will only be imposed if the task is recommended.

The routing subproblem is defined for each (k, m) tuple; the total number of routing subproblems is thus $\sum_{k \in K} |M_k|$. For each (k, m) tuple, let $f_k^m(\boldsymbol{\lambda}_k^m)$ be the corresponding routing subproblem, where

$$\boldsymbol{\lambda}_k^m = \{\ldots, \lambda_{i-1,k}^m, \lambda_{i,k}, \lambda_{i+1,k}^m, \ldots\},$$

and $f_k^m(\boldsymbol{\lambda}_k^m)$ is defined as:

$$\min - \sum_{i \in N_t} \lambda_{ik}^m \sum_{j \in N} x_{ijk}^m, \quad (15)$$

with constraints (3) – (8).

The Lagrangian dual problem is solved by using a standard subgradient descent algorithm as follows. Given a $\boldsymbol{\lambda}$ vector (initialized to zeros), we solve all dual subproblems; obtain the dual objective function value, and use a primal extraction procedure to obtain primal objective function value. The dual objective function value of our problem can be computed by simply summing up all objective values from the subproblems (i.e., (13) and (15)). The corresponding primal solution can be obtained by inserting the dual solutions ($\{x_{ijk}^m\}$) back into the original problem (i.e. optimizing (10) subject to the same set of primal constraints (2) and (11).

The Lagrangian multipliers are updated according to the function below:

$$\lambda_{ik,t+1}^m := \lambda_{ik,t}^m + \alpha_t (y_{ik} - \delta_{ik}^m - \sum_{j \in N} x_{ijk}^m).$$

We perform this procedure iteratively and terminate if the duality gap is less than certain threshold or we have exhausted allocated computational time.

The update step size α_t is defined to be adaptive to both the duality gap and the solution quality (the magnitude of constraint violation):

$$\alpha_t = \frac{\mu_t \left(F^* - L(\boldsymbol{\lambda}_t) \right)}{\sum_{i \in N_t, k \in K, m \in M_k} \left(y_{ik} - \delta_{ik}^m - \sum_{j \in N} x_{ijk}^m \right)^2},$$

where $(\{x_{ijk}^m\}, \{y_{ik}\}, \{\delta_{ik}^m\})$ are obtained by solving dual subproblems, F^* is the best primal value seen so far, and $L(\boldsymbol{\lambda}_t)$ represents dual value obtained in iteration t. μ_t is in the range of $(0, 2]$, and is defined as:

$$\mu_t = \begin{cases} 2, & t = 0, \\ 0.5\ \mu_{t-1}, & \text{if } L(\boldsymbol{\lambda}_t) \text{ fails to increase for } \kappa \text{ iter.} \end{cases}$$

4.1 LR Variants

The performances of our LR heuristics depend on how subproblems are solved. The assignment subproblem can be solved exactly and efficiently using a greedy algorithm (it's provably optimal since each task can be recommended to a particular agent purely based on the contribution to the objective function value, and is independent of other tasks and other agents; the proof is omitted in the interest of space). The routing subproblem, on the other hand, can not be solved so efficiently, and are the major performance bottleneck. To investigate the trade-off between the optimality and the time performance, we define the following two LR variants based on how the routing subproblems are solved:

- **LR-Exact**: Routing subproblems are solved exactly by pure enumeration. Pure enumeration is the preferred approach in most instances since with reasonable detour limit (say up to 30%), the number of feasible tasks will be small enough such that pure enumeration will outperform regular routing algorithm; a threshold can be set for the solver to switch to regular router if number of feasible tasks is too large.
- **LR-Greedy**: Routing subproblems are solved using a simple greedy heuristic: an agent starts with the routine route m, and repeatedly try to evaluate the gain of inserting one of the remaining tasks into all potential slots; of all possible (task, slot) combinations, the best is chosen (thus greedy). There is no optimality guarantee, but it can solve routing subproblems very efficiently.

5 Experiment Results

The ILP model and two corresponding LR heuristics have been introduced in Sections 3 and 4, with these, we have addressed the first major contribution of the paper: modeling task recommendation under route uncertainty using ILP model and come up with efficient heuristics. What remained to be shown is the quality of LR heuristics, and the benefits of modeling routine route uncertainty in a large-scale mobile crowdsourcing operation. The numerical experiments in this section are designed to achieved these two goals.

More specifically, we will investigate the performance of our LR heuristics from the following two perspectives:

- **Performances of LR heuristics**: We will compare the two LR heuristics against the optimum obtained by solving ILP model exactly. Both solution quality and computational time will be measured. This evaluation will be limited to only small instances due to the complexity of solving our ILP model exactly.
- **Deterministic versus stochastic models**: We will also compare the performance of LR heuristics against two deterministic planning approaches. One is the *push-based* deterministic model proposed by [Chen *et al.*, 2014]; the other is the *pull-based* proximity approach that emulates current best practices. We will use city-scale network topology to perform this comparison.

5.1 Performances of LR heuristics

The purpose of this evaluation is to compare LR-Exact and LR-Greedy to the exact ILP model, both in terms of solution

quality and computational time. Test instances are generated randomly with parameters (K, N_t, N), where K refers to the number of agents, N_t refers to the number of task nodes, and N refers to the total number of nodes in the network. Each agent is assumed to have two routine route candidates, where all routes have 5 nodes and are selected with equal probability. The coordinates of all nodes are generated uniformly randomly on a grid network. The distance between all pairs of nodes are Euclidean distance.

(K, N_t, N)	ILP	LR-Exact		LR-Greedy	
	time	gap	time	gap	time
(2,4,40)	0.8s	0%	0.09s	0%	0.05s
(4,8,80)	22.9s	0%	0.2s	4.12%	0.06s
(8,16,80)	6558s*	0.06%	14.8s	0.06%	0.14s

Table 1: LR heuristics vs. ILP: on both quality and time.
*: We cut off CPLEX solver as the optimality gap is only 0.06%.

Our evaluation is summarized in Table 1 above. The gap is percentage from the optimum obtained via solving ILP model exactly. From these small testing instances, we can see that both LR heuristics can produce close-to-optimum solutions very quickly. But examining the results closer, we can see that the efficiency of LR-Exact and LR-Greedy can potentially be different by one to two orders of magnitude.

5.2 Deterministic vs. Stochastic Models

To empirically quantify the benefits of generating recommendations considering routine route uncertainties, we introduce the following two deterministic baselines to compare with:

- **Deterministic-ILS**: Following the deterministic model and the iterated local search (ILS) heuristic proposed by [Chen *et al.*, 2014], we designate each agent's routine route to be the route with highest probability. We denoted this baseline as **DILS**.
- **Proximity-Based Approach**: This heuristic emulates how most pull-based mobile crowdsourcing platforms work nowadays. At each decision epoch, the agents who are available will be given the opportunity to pick desired tasks based on proximity. We denoted this baseline as **Proximity**.

The network used in this evaluation is based on the actual public transit network in Singapore (4,296 nodes and 10,129 edges), which contains all stops from the metro and bus services. All-pair-shortest distance matrix is computed a priori. To reflect the heterogeneous travel patterns of agents, we include two types of agents: 80% of *normal agents* who compute back and forth between fixed locations (e.g., home and office), and 20% of *freelancers* whose routine routes have randomly chosen origin and destination nodes. For both agent types, the origin nodes are randomly picked from the non-central zones, while the end nodes are from the central zones (reflecting a commuting pattern from "home" to "office"). Two routes are constructed for each agent as follows: (i) the shortest path between the chosen origin and destination, and (ii) the path with the least number of stops. The probability distribution over agent k's two routes follows Bernoulli distribution with parameter α_k, where α_k is sampled uniformly from $(0, 1)$. Locations of tasks are generated according to the distribution (p_r, p_h), where p_r and p_h refer to the ratio of tasks in the non-central and the central zones. For the synthetic instances, half are generated with (p_r, p_h) equals: (60%, 40%), while the rest with (40%, 60%). Each task is associated with a fixed utility value of 100.

	Estimate	LR-E	LR-G	DILS	Proximity
Detour	Bound	Gap	Gap	Gap	Gap
10%	54.5%	-0.6%*	0.6%	13.2%	15.3%
20%	77.4%	-0.9%*	0.1%	10.4%	12.2%
30%	91.9%	0.1%	1.1%	7.1%	8.6%

Table 2: LR heuristics vs. deterministic baselines.

The results are categorized using different detour limits and (K, N_t, N) tuples, where 20 synthetic instances are randomly generated for each category using the above scheme. To evaluate the performance of all competing approaches in a particular instance, 1000 routine route realizations are sampled. Table 2 shows the reduction in the average task completion ratio (compared to an estimated upper bound) achieved by these methods for the tuple (20,30,160). The estimated upper bound is computed by assuming full knowledge of route realizations and then solve the deterministic problem using DILS. This is why in some cases (*), the LR-Exact approach actually outperforms the estimated bound. From Table 2 we can clearly see the advantage of considering route uncertainties. Due to the space limit, we cannot comprehensively list all experiment results, however, in all instances we evaluated, LR-based approaches outperform deterministic alternatives, and the advantages of LR heuristics increase further as we tighten detour limits.

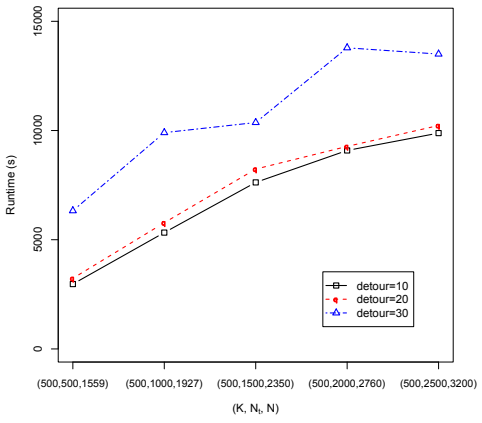


Figure 2: Runtime(s) when N_t is increased.

In terms of scalability, LR-Exact does not scale well. Without parallelization, LR-Exact takes more than 9 hours to solve $(100, 200, 600)$ instances, which is of moderate size. LR-

Greedy, on the other hand, is much more scalable. In Figure 2, we fix K at 500 and increase N_t from 500 to 2500; in Figure 3, we fix N_t at 1000 and increase K from 200 to 1000. From both figures, we empirically observe that runtime scales linearly in K and N_t.

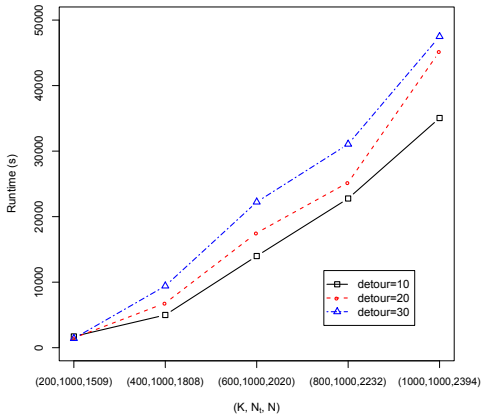


Figure 3: Runtime(s) when K is increased.

6 Conclusions

In this paper, we investigate a "push-based" paradigm for large-scale mobile crowdsourcing, where a centralized engine recommends tasks for a large pool of workers, while taking into account the inherent probabilistic uncertainty about their future trajectories. A stochastic integer linear program is developed to find assignments that maximize the cumulative expected utility. Subsequently, to develop a solution that can scale to city-scale scenarios, we exploit the separable problem structure and apply the Lagrangian relaxation approach. Experiments using realistic movement traces over Singapore's public transport network show that our approach can find significantly better solutions (10-12% higher task completion rates) than the deterministic baselines, and are especially useful for expected real-world situations, where the available worker pool is limited and workers are likely to accept only modest task-related detours.

References

[Alt *et al.*, 2010] Florian Alt, Alireza Sahami Shirazi, Albrecht Schmidt, Urs Kramer, and Zahid Nawaz. Location-based crowdsourcing: extending crowdsourcing to the real world. In *6th Nordic Conference on Human-Computer Interaction: Extending Boundaries*, pages 13–22. ACM, 2010.

[Boutsis and Kalogeraki, 2014] Ioannis Boutsis and Vana Kalogeraki. On task assignment for real-time reliable crowdsourcing. In *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*, June 2014.

[Chao *et al.*, 1996] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

[Chen *et al.*, 2014] Cen Chen, Shih-Fen Cheng, Aldy Gunawan, Archan Misra, Koustuv Dasgupta, and Deepthi Chander. TRACCS: Trajectory-aware coordinated urban crowd-sourcing. In *2nd AAAI Conference on Human Computation and Crowdsourcing*, pages 30–40, 2014.

[Howe, 2006] Jeff Howe. The rise of crowdsourcing. *Wired Magazine*, 14(6):1–4, 2006.

[Kanhere, 2011] Salil S Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *12th IEEE International Conference on Mobile Data Management*, pages 3–6, 2011.

[Kazemi and Shahabi, 2011] Leyla Kazemi and Cyrus Shahabi. A privacy-aware framework for participatory sensing. *ACM SIGKDD Explorations Newsletter*, 13(1):43–51, 2011.

[Kazemi and Shahabi, 2012] Leyla Kazemi and Cyrus Shahabi. Geocrowd: enabling query answering with spatial crowdsourcing. In *20th International Conference on Advances in Geographic Information Systems*, pages 189–198. ACM, 2012.

[Musthag and Ganesan, 2013] Mohamed Musthag and Deepak Ganesan. Labor dynamics in a mobile micro-task market. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 641–650, 2013.

[Rula *et al.*, 2014] John P. Rula, Vishnu Navda, Fabián E. Bustamante, Ranjita Bhagwan, and Saikat Guha. No "one-size fits all": Towards a principled approach for incentives in mobile crowdsourcing. In *15th Workshop on Mobile Computing Systems and Applications*, pages 3:1–3:5, 2014.

[Sadilek *et al.*, 2013] Adam Sadilek, John Krumm, and Eric Horvitz. Crowdphysics: Planned and opportunistic crowdsourcing for physical tasks. In *7th International AAAI Conference on Weblogs and Social Media*, pages 536–545, 2013.

[Tricoire and Hartl., 2010] Martin Romauch Karl F. Doerner Tricoire, Fabien and Richard F. Hartl. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers and Operations Research*, 37(2):351–367, 2010.

[Tsiligirides, 1984] Theodore Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[Vansteenwegen *et al.*, 2011] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1 – 10, 2011.

[Yang *et al.*, 2012] Dejun Yang, Guoliang Xue, Xi Fang, and Jian Tang. Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing. In *18th Annual International Conference on Mobile Computing and Networking*, 2012.