Transportation Research Part E 93 (2016) 279–293

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Towards enhancing the last-mile delivery: An effective crowd-tasking model with scalable solutions

Yuan Wang [a], Dongxiang Zhang [c,*], Qing Liu [b], Fumin Shen [c], Loo Hay Lee [a]

[a] Department of Industrial and Systems Engineering, National University of Singapore, Singapore 119260, Singapore
[b] Department of Computer Science, School of Computing, National University of Singapore, 117417, Singapore
[c] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China

ARTICLE INFO

Article history:
Received 8 February 2016
Received in revised form 9 June 2016
Accepted 12 June 2016
Available online 1 July 2016

Keywords:
Crowdsourcing
Last-mile delivery
Minimum cost flow

ABSTRACT

In urban logistics, the last-mile delivery from the warehouse to the consumer's home has become more and more challenging with the continuous growth of E-commerce. It requires elaborate planning and scheduling to minimize the global traveling cost, but often results in unattended delivery as most consumers are away from home. In this paper, we propose an effective large-scale mobile crowd-tasking model in which a large pool of citizen workers are used to perform the last-mile delivery. To efficiently solve the model, we formulate it as a network min-cost flow problem and propose various pruning techniques that can dramatically reduce the network size. Comprehensive experiments were conducted with Singapore and Beijing datasets. The results show that our solution can support real-time delivery optimization in the large-scale mobile crowd-sourcing problem.

1. Introduction

Cargo transportation via rail networks and container ships is considered as the most cost-effective manner in urban logistics. However, when the goods arrive at the high-capacity warehouses, they must then be transported to the final destinations. This last leg of the supply chain is less efficient and comprises up to 28% of the total delivery cost.[1] Worse still, most consumers are not present when the deliveries are made. The unattended parcels may require multiple times of attempt-delivery and have become a significant issue among logistic companies.

To mitigate the situation, the concept of pop-station (pick-own-parcel station) has been adopted by some logistic companies such as Singapore Post.[2] The idea is that the parcels are directly delivered to the pop-stations. Then, consumers will be notified and collect their own parcels via mobile apps. If a parcel is not collected within 5 days,[3] it is considered as a failed delivery. To ensure that the model works, it requires expensive infrastructure costs because a large number of pop-stations have to be built to benefit residents in different areas of the city and minimize their walking distance for self-collection. In addition, the problem of reducing the parcel turnaround time for fast delivery is not well solved in this model. Many parcels may be kept in the lockers of pop-stations for several days.

* Corresponding author at: School of Computer Science and Engineering, University of Electronic Science and Technology of China, No. 2006, Xi Yuan Road, Chengdu, Sichuan 611731, China.
E-mail address: zhangdongxiang37@gmail.com (D. Zhang).

[1] https://en.wikipedia.org/wiki/Last_mile.
[2] https://www.mypopstation.com/.
[3] https://www.mypopstation.com/faqs#faq-02-06.

http://dx.doi.org/10.1016/j.tre.2016.06.002
1366-5545/

In this paper, we investigate how to utilize the power of crowd-workers to enhance the last-mile delivery. In particular, we treat the delivery job of each parcel from its pop-station to the consumer as a crowd-task. A certain amount of money will be rewarded to a worker for the delivery according to the additional travel cost from his/her historical trajectory patterns. Our objective is to assign all the parcels in the pop-stations to the most convenient workers so as to minimize the total reward paid by the logistic companies. The underlying principles are similar to other sharing-economy applications to maximize the resource (crowd-worker in this application) utilization. With the availability of a large pool of citizen crowd-workers, the infrastructure expenses can be cut down as the crowd-delivery model requires much fewer number of pop-stations than the model of self-collection. In addition, both the parcel turnaround time and failure rate can be significantly reduced because the crowd-workers are more active in collecting the parcels and delivering them to the consumers.

Since there could be a huge number of parcels and workers in an urban city, our crowd-delivery model is essentially a large-scale assignment optimization problem. Our solution is to model it as a network min-cost flow problem and use it as the baseline for performance evaluation. Then, we propose various effective pruning strategies that can significantly reduce the network size. Comprehensive experiments were conducted with Singapore and Beijing datasets. The results show that after the network reduction, the performance achieves a speedup by 2–3 orders of magnitude. It takes less than 10 s to find the optimal assignment of 2000 parcels to a pool of 500,000 crowd-workers.

To sum up, the contributions of the paper include:

- We formulate an interesting crowd-logistics optimization problem that utilizes a large pool of citizen workers to perform the last-mile delivery.
- We formally show that the proposed model is equivalent to the network min-cost flow problem.
- We propose three types of pruning rules that can significantly reduce the network size, and hence improve the performance.
- We conduct comprehensive experiments using Singapore and Beijing datasets to verify the efficiency of our proposed methods.

The rest of the paper is organized as follows. We present the problem definition in Section 2 and review related literature in Section 3. The problem is formally reduced to a minimum cost flow problem in Section 4. Various pruning techniques are proposed in Section 5 to significantly reduce the network size. We conduct an extensive performance study in Section 6 to evaluate the performance of our proposed solutions. Section 7 concludes the paper with future work.

2. Problem definition

2.1. Background

In our crowd-delivery model, there are a bunch of pop-stations distributed around the city and a large pool of workers who are ready to accept the delivery tasks from pop-stations to the consumers' addresses. As shown in Fig. 1, we can split the city into voronoi cells according to the locations of pop stations. The logistic companies only need to be focused on the scheduling optimization of delivering parcels to the pop-stations. Intuitively, each parcel will be sent to the nearest pop-station according to its consumer's address. In other words, the final delivery address and the associated pop-station will be located within the same voronoi cell. Thereafter, the parcels at the pop-stations will be assigned to the crowd-workers and eventually reach the consumers.

When a worker accepts a task via a mobile app (similar to an Uber driver accepting a riding request), he/she can collect the parcel from the pop-station with a one-time-password. After that, the system notifies the consumer that the parcel has been taken, and starts tracking the real-time locations of the worker. When the parcel is safely delivered, a confirmation message is sent from the consumer and this transaction completes. To improve the system reliability and service quality, the identifies of crowd-workers have to be verified, which is common in apps like Uber and GrabTaxi.

We also note that UberRush[4] has provided parcel delivery service, using the drivers as workers. The major difference is that UberRush processes on-demand delivery requests and sends them to nearby drivers. It's one-time processing and there is no complex optimization issue. In contrast, we focus on the last-mile delivery to improve the efficiency of the whole supply chain. Our method is to leverage a large pool of crowd-workers to finish the last leg of delivery with any possible transportation means. In other words, we treat the transportation means as a black-box and these crowd-workers can walk, take a bus/train or drive a car to complete the delivery task.

When compared with conventional collaborative or synchronized approaches, as shown in Fig. 2, our system demonstrates the superiority from the following perspectives.

[4] https://rush.uber.com/how-it-works.

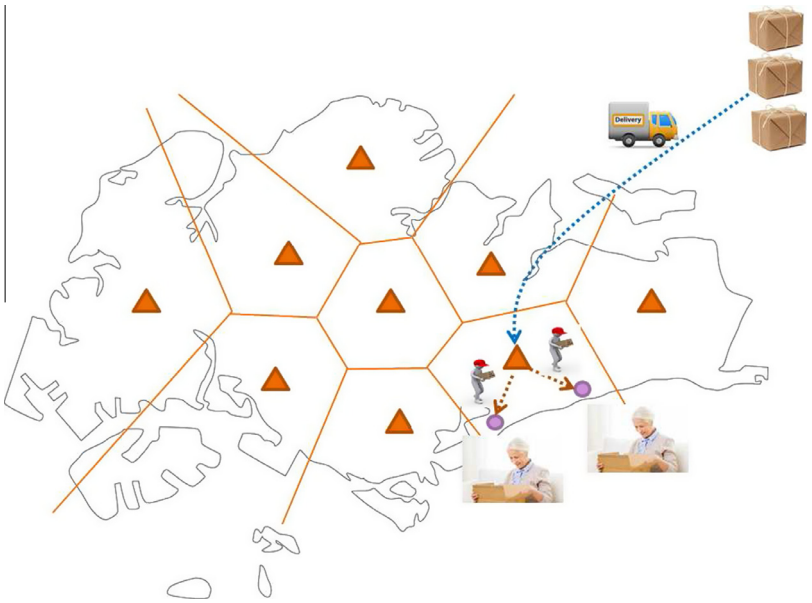

Fig. 1. Crowdsourcing for the last-mile delivery.

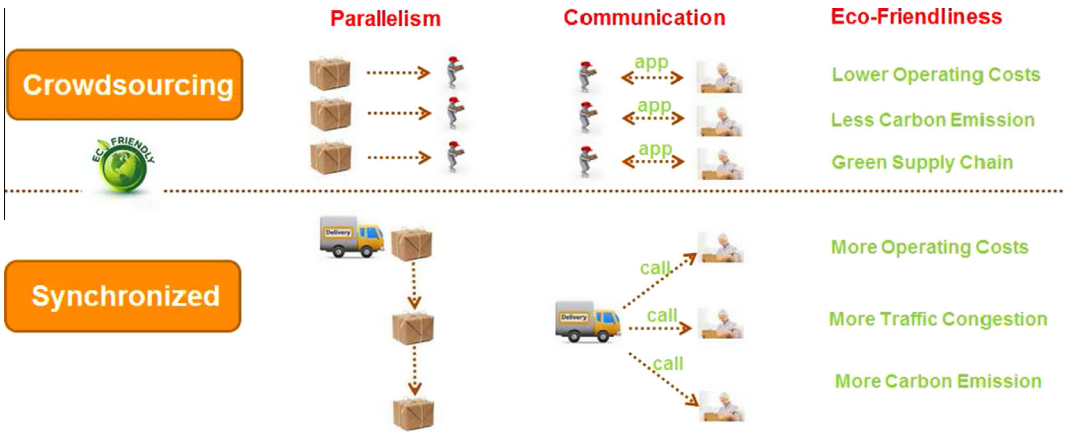

Fig. 2. Comparison of the crowd-delivery with the conventional synchronized-delivery.

- **Parallelism**. It is obvious that the number of available workers in the crowd-delivery model is much higher than the number of delivery vehicles in the synchronized-delivery model, incurring much higher parallelism in job execution. Even though the collaborative model can improve the resource utilization, each vehicle still has to handle a large number of parcels per day and requires elaborate planning.
- **Communication**. In our model, each worker only handles a small number of parcels and the communication channel between delivery workers and parcel receivers are much more effective. Even if a parcel is unattended when the receiver is not available, its effect on the subsequent delivery is minimized. In contrast, when such case occurs in the synchronized delivery, all the subsequent deliveries by the same vehicle may be delayed.
- **Eco-Friendliness**. Our crowd-delivery model is considered as more eco-friendly because it can help reduce the operation costs and carbon emission. The impact to conventional urban logistics could be similar to that of Uber to the conventional Taxi industry.

2.2. Model and objective function

Intuitively, a parcel should be sent to a worker with the minimum additional efforts. For example, suppose Alice is a clerk in a shopping mall and lives at Andersen street. If there is parcel in a pop-station located in the same shopping mall and to be

delivered to Andersen street, then Alice is a very good candidate to take the task. Based on the criterion, we formulate our crowd-delivery model with the following simplified assumptions:

- Each worker is associated with a travel pattern represented by $A \rightarrow B$. Take Alice as an example, her motion pattern would be "shopping mall→Andersen street".
- The reward for the delivery of a parcel is dependent on the additional efforts for a crowd-worker. For simplicity, we use additional travel distance as an estimation. For example, when a parcel p_i is delivered from pop-station s to destination t by a worker w_j with a travel pattern $A \rightarrow B$, its cost $c(p_i, w_j)$ is measured by the additional travel distance:

$$c(p_i, w_j) = d(A, s) + d(s, t) + d(t, B) - d(A, B) \tag{1}$$

  where $d(\cdot, \cdot)$ refers to the Euclidean distance between two locations; $d(A, s) + d(s, t) + d(t, B)$ is the distance incurred for the delivery; and $d(A, B)$ is the travel distance if the worker does not take the task.
- For simplicity, we assume that the delivery of different parcels is independent. Then, the total payment for a worker is the sum of each delivery cost.
- We assume that each worker has a delivery capacity C to determine the maximum number of parcels that he/she can carry.
- The time constraints of consumers are not taken into account in the model. We simply assume that the delivery time is negotiated between the consumer and the assigned worker.

With these simplified assumptions, we can build a large-scale crowd-delivery model that can be solved with feasible computational cost. Our objective is to assign all the parcels to the best candidate workers with the minimum expense. Let $x(p_i, w_j) = 1$ denote parcel p_i is assigned to w_j and $x(p_i, w_j) = 0$ for the opposite case. Formally, our model minimizes

$$\sum_{p_j} \sum_{w_j} c(p_i, w_j) x(p_i, w_j) \tag{2}$$

subject to

$$\sum_{p_i} x(p_i, w_j) \leqslant C \tag{3}$$

$$\sum_{w_j} x(p_i, w_j) = 1 \tag{4}$$

Constraint (3) restricts the number of parcels assigned to a worker must be no greater than C. Constraint (4) means each parcel must be assigned to one worker.

The crowd-delivery system is often much more complex in reality. For example, when a worker refuses to take the assigned task, the system needs to find another replacement or offer higher price. This process may require multiple rounds of communication with the workers before the task can be finally assigned. Other challenging issues include better pricing strategy and system service reliability. All these issues are interesting research problems for the crowd-based last-mile delivery. Our work in this paper can be viewed as initial efforts to formulate the model with simplified assumptions. By solving the model, we obtain a good assignment for each parcel. These assignments can be further refined by subsequent optimization steps, which are considered as our future works.

3. Related work

In this section, we conduct a literature review about last-mile delivery and mobile crowdsourcing.

3.1. Last-mile delivery

In recent years, there have been several efforts attempting to improve the efficiency of the last-mile delivery. We summarize the applied strategies into two major categories. In the first category, the concept similar to the pop-station was adopted. It can effectively solve the headache of unattained parcels when the consumer is not home during the delivery. The parcels are stored in pop-stations with lockers in various size. The consumers are then notified to collect the parcels by themselves with a one-time-password. For instance, in DellAmico and Hadjidimitriou (2012), Modular BentoBox System (M-BBX) was proposed to deliver goods to bentobox where they are stored until the customer picks them up. Logistic operators such as DHL and Austrian Post also adopted the idea and introduced the Packstation system and the Post.24 Parcel Machines respectively.

In the second category, the efficiency was improved by better consolidation and synchronization of existing resources. The main idea is that multiple logistic companies can share the delivery vehicles and staff to improve the resource utilization rate. It is similar to the codeshare agreement among different airline companies to reduce the operation expenses. In de Souza et al. (2014), the authors proposed the concept of collaborative urban logistics and emphasized multi-party collaboration from the perspective of Singapore Logistics. Their objective is to extend and optimize the respective parties

resource portfolios and to reinforce their own market position via better resource coordination and data harmonization. In Liakos and Delis (2015), an interactive freight-pooling service was proposed to reduce the undesirable effects and the cost of freight transport in urban areas. The optimization model applied is similar to the capacitated vehicle routing problem (Dantzig and Ramser, 1959) with the goal of synchronizing supply chains of multiple shippers and transferring the same number of goods using significantly less resources. In Petrovic et al. (2013), the authors studied how to overcome the communication barrier by allowing recipients to use smartphones to open a communication channel to logistics providers. In Handoko et al. (2014), the last-mile delivery tasks are outsourced to urban consolidation centers and a profit-maximizing auction mechanism was proposed to determine which demands are to be served.

3.2. Mobile crowdsourcing

In recent years, mobile crowdsourcing has been a hot research topic in various task domains due to the provenance of smartphones and the booming of location-based services. When combining crowdsourcing with mobile devices, a large number of interesting applications have emerged. For instance, GeoCrowd (Kazemi and Shahabi, 2012) assigns nearby tasks to each worker. The objective is to maximize the overall number of assigned tasks because each task is assigned to one work and all the constraints have to be satisfied. Three types of greedy heuristics were proposed to solve the optimization problem. In Su et al. (2014) and Zhang et al. (2014), crowd workers are used for travel path recommendation. The research challenge is how to effectively summarize the feedback from all the workers and provide constructive suggestion. An entropy-based metric was proposed to capture the path selection hardness and the problem is transferred to an optimization problem based on the new metric. In Cheng et al. (2015), an interesting problem to piece together a full story from different crowd workers is studied.

Systems like CrowdSC (Benouaret et al., 2013) and gMission (Chen et al., 2014b) were developed towards building smart cities with large-scale citizen participation. Ilarri et al. (2014) overviews the current status of research in the field of collaborative sensing for urban transportation such as parking spaces, traffic, and trajectories. Privacy issues was addressed in Pournajaf et al. (2014) to protect the worker's privacy. The most related work to our crowd-delivery model is the TRACCS system (Chen et al., 2014a). The crowd-tasking platform assigns a sequence of tasks to each worker, taking into account their trajectory patterns. It formulates the task assignment as an optimization problem with the objective of maximizing the total payoff from all assigned tasks. An Integer Linear Programming model is presented, which unfortunately can handle very small-scale problems. To improve the efficiency, greedy heuristics, without theoretical guarantee, were proposed. Its experimental study only evaluates up to 1000 agents. In comparison, our crowd delivery platform requires a solution that can handle million-scale workers in order to become profitable as more crowd workers can help reduce the price per task.

4. Minimum cost flow model

In this section, we formulate the optimization of the proposed crowd-delivery model as a minimum cost flow problem. The notations frequently used in this paper are listed in Table 1 for quick reference.

4.1. Background

The min-cost flow model is a fundamental problem in the network flow domain (Ahuja et al., 1993). Given a directed graph $G = (V, E)$, each arc $e \in E$ is associated with a capacity u_e and a unit transportation cost c_e. Each node v furthermore has a number b_v representing its supply/demand. If $b_v > 0$, node v is a *surplus node*; if $b_v < 0$, node v is a *demand node* with a demand value of $-b_v$; and if $b_v = 0$, node v is called a *transshipment node*. It is required that there are balances between the supply and demand, i.e., $\sum_{v \in V} b_v = 0$.

Let $N^+(v)$ denote the out-neighborhood of v and $N^-(v)$ denote the in-neighborhood of v. The min-cost flow problem considers how to supply the demand nodes from the supply nodes by a flow in the cheapest possible way. The decision variables of the problem are the arc flows, which are denoted by f_e. Then, the problem can be formalized as an optimization problem with the objective

$$\min \sum_{e \in E} c_e f_e \tag{5}$$

subject to

$$f_e \leqslant u_e \tag{6}$$

$$b_v = \sum_{e \in N^+(v)} f_e - \sum_{e \in N^-(v)} f_e \tag{7}$$

Fig. 3 depicts a toy example of the min-cost flow problem. There are 5 nodes and 8 arcs in the graph. Among the nodes, v_1 is a surplus node because $b_{v_1} = 30$. v_2 and v_3 are transshipment nodes and their values of b_v are equal to 0 and not displayed in the figure. v_4 and v_5 are demand nodes. The total amount of demand is equal to the supply. Each arc is associated with a capacity u_e and a unit transportation cost c_e, represented in the form of c_e/u_e in the figure. Our goal is to determine a least

Table 1
Notation table.

p_i	A parcel
w_j	A crowd worker
s_k	A pop station
P	The number of parcels
W	The number of workers
S	The number of pop stations
C	The capacity of a worker
$c(p_i, w_j)$	The cost of assign a parcel p_i to worker w_j
$x(p_i, w_j)$	$= 1$ if p_i is assigned to w_j; otherwise, $x(p_i, w_j) = 0$
$d(\cdot, \cdot)$	The Euclidean distance between two locations
$p_i \leftarrow w_j$	Assign parcel p_i to worker w_j

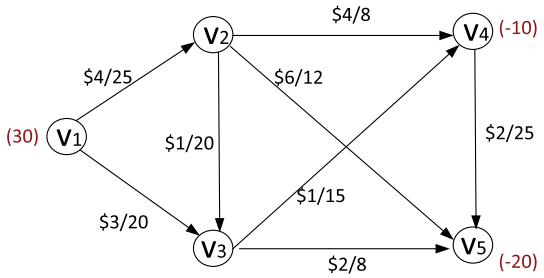

Fig. 3. An example of min-cost flow problem.

cost shipment through the network in order to meet the demands at $\{v_4, v_5\}$ from the surplus v_1. More specifically, we need to determine the flow f_e in each arc satisfying Constraints (6) and (7) such that the total cost is minimized. The optimal flow for this example is $\{f_{12} = 10, f_{13} = 20, f_{23} = 3, f_{24} = 0, f_{25} = 7, f_{34} = 15, f_{35} = 8, f_{45} = 5\}$ with the total cost of 186.

4.2. Reduction to the min-cost flow model

To reduce our crowd-delivery model to the min-cost flow problem, we build a network as shown in Fig. 4. The top and bottom are two dummy nodes s and t, represented in dashed circles: s is the only surplus node and its b_s is set to the number of parcels P, meaning the total amount of supply is P; and t is the only demand node, with b_t set to $-P$ for the balance between supply and demand. It is worth noting that the concept of suppliers in the network min-cost flow model is not equivalent to the delivery capacity of all the available workers. Instead, we assume that the total capacity of workers is higher than the number of parcels and we can always find a solution based on the optimization model. The remaining nodes, including parcels and crowd-workers, serve as the transshipment nodes.

Three types of arcs, denoted by $s \to p_i, p_i \to w_j$ and $w_j \to t$ respectively, are incorporated in the network. The first type $s \to p_i$ is from the surplus node to the parcels, with capacity $u_e = 1$ and transportation cost per unit $c_e = 0$. The second type $p_i \to w_j$ is from parcels to workers. The capacity is set to 1, meaning each parcel can be assigned to at most one worker and the cost of the assignment is set to $c(p, w)$, measured by the additional travel distance. The third type of arcs $w_j \to t$ is from workers to the demand node. The capacity is set to the maximum number of parcels that a worker can carry, which is C in our model.

Let Δ_{opt} denote the optimal solution to the min-cost flow problem in Fig. 4. It essentially contains the values of f_e for all the arcs. To show that our crowd-delivery model can be reduced to the min-cost flow problem, we first prove the following lemma.

Lemma 1. *$f_{s \to p_i} = 1$ for all p_i.*

Proof. Based on Constraint (6), we know that $f_{s \to p_i} \leqslant 1$. Suppose there exists an arc $s \to p_i'$ such that $f_{s \to p_i'} < 1$. Then, $\sum_{e \in N^+(s)} f_e - \sum_{e \in N^-(s)} f_e \leqslant P - 1 + f_{s \to p_i'} < P = b_s$, which violates Constraint (7). Therefore, all the $f_{s \to p_i}$ must be set to 1. □

Next, we prove that when we set $x(p_i, w_j) = f_{p_i \to w_j}$, the objective function in the crowd-delivery model is the same as the cost function in the network flow problem.

Lemma 2. *If $x(p_i, w_j) = f_{p_i \to w_j}$, then $\sum_{e \in E} c_e f_e = \sum_{p_i} \sum_{w_j} c(p_i, w_j) x(p_i, w_j)$*

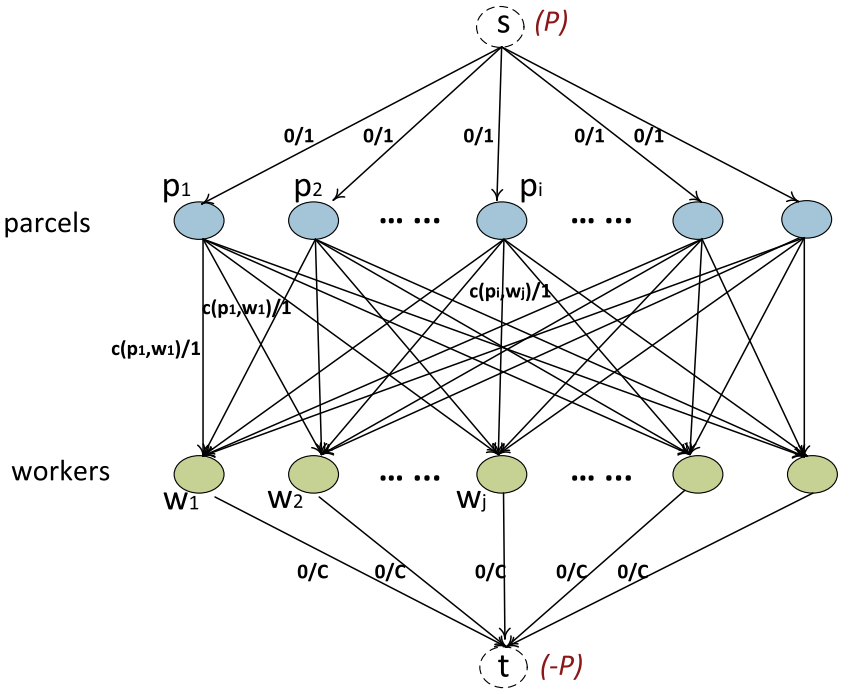

Fig. 4. Reduction to min-cost flow model.

Proof. Since the costs for arcs $s \to p_i$ and $w_j \to t$ are both set to 0, we have $\sum_{e \in E} c_e f_e = \sum_{p_i} \sum_{w_j} c_{p_i \to w_j} f_{p_i \to w_j}$. Since $x(p_i, w_j) = f_{p_i \to w_j}$, we finish the proof. □

Finally, we show that Constraints (3) and (4) are also satisfied based on the mapping between $x(p_i, w_j)$ and $f_{p_i \to w_j}$.

Lemma 3. *If* $x(p_i, w_j) = f_{p_i \to w_j}$, *then* $\sum_{p_i} x(p_i, w_j) \leqslant C$.

Proof. Based on Constraint (7), we have $b(w_j) = f_{w_j \to t} - \sum_{p_i} f_{p_i \to w_j} = 0$. Based on Constraint (6), we have $\sum_{p_i} x(p_i, w_j) = \sum_{p_i} f_{p_i \to w_j} = f_{w_j \to t} \leqslant C$. □

Lemma 4. *If* $x(p_i, w_j) = f_{p_i \to w_j}$, *then* $\sum_{w_j} x(p_i, w_j) = 1$.

Proof. Based on Constraint (7), we have $b(p_i) = \sum_{w_j} f_{p_i \to w_j} - f_{s \to p_i} = 0$. Then, according to Lemma 1, $\sum_{w_j} x(p_i, w_j) = \sum_{w_j} f_{p_i \to w_j} = f_{s \to p_i} = 1$. □

Therefore, when we find a solution Δ_{opt} for the min-cost flow problem, the cost is also the minimum value of our objective function in the crowd-delivery model as long as we set $x(p_i, w_j) = f_{p_i \to w_j}$.

4.3. Implementation

After reducing our crowd-delivery model to the min-cost flow problem, we can adopt any standard network flow techniques to solve our problem. A general solution to the problem is linear programming, since we optimize a linear function, and all constraints are linear. There are several other tailored algorithms, such as cycle-canceling algorithm, cost-scaling algorithm, successive shortest path algorithm and network simplex algorithm, have also been proposed to improve the performance. Readers can refer to Ahuja et al. (1993) for a comprehensive survey. In our implementation, we adopt the network simplex algorithm (Florian and Lebeuf, 1997; Kennington and Helgason, 1980) provided by the LEMON library[5] as the baseline solution for the crowd-delivery problem in this paper. The simplex algorithm is selected because it was experimentally evaluated in Király and Kovács (2012) as the fastest implementation for the min-cost flow problem.

[5] http://lemon.cs.elte.hu/trac/lemon.

The computation cost consists of two parts: the network construction time and the model solving time. To construct the network, we need to traverse each pair (p_i, w_j) and calculate the additional travel distance $c(p_i, w_j)$ as the unit cost in the network. The complexity is $O(P \cdot W)$ where P is the number of parcels and W is the number of workers. For the model solver, the complexity of applying the network simplex algorithm in our problem is $O(|V||E|^2 \cdot C \cdot P)$, where $|V| = P + W$ is the number of nodes in the network, $|E| = P \cdot W$ is the number of arcs and C is the maximum capacity.

5. Pruning strategies

The complexity of the above baseline solution is dominated by $|E|^2$, which could be very huge when there are many workers and parcels in the model. In this section, we propose various pruning techniques with the purpose of reducing $|E|$.

5.1. Rule 1: Cost-based pruning

Our first pruning strategy consists of two steps. In the first step, we propose a greedy method to quickly assign the parcels to the workers with a relatively small cost. This cost, denoted by δ_g, acts as the upper bound for the optimal solution. In the second step, given a pair of parcel p_i and worker w_j, we estimate the lower bound for any solution which assigns p_i to w_j. If the lower bound is even higher than δ_g, we can remove the arc $p_i \rightarrow w_j$ from the network because the cost of the optimal solution should be no greater than δ_g.

Our greedy algorithm is presented in Algorithm 1. In the initialization steps, we build an inverted list for each parcel p_i. The elements in the list are in the form of $(w_j, c(p_i, w_j))$ and sorted in ascending order of $c(p_i, w_j)$. Thus, the promising assignments will be put at the beginning of the list. We also construct a set M with all the sorted lists and initialize the value of δ_g to 0. Then, the algorithm starts iterating among the sorted lists to find the assignment (p_i, w_j) with the minimum cost. If the worker w_j still has capacity to accept the task, we consider it as one assignment in the greedy solution and update the total cost of δ_g. Since we have finished the assignment of parcel p_i, we remove its sorted list L_i from M. If the worker w_j has been assigned with C parcels, we cannot assign p_i to w_j and need to find other candidate in the sorted list. We continue the iterations until all the parcels have been assigned. At this moment, M becomes empty and δ_g is the cost of our greedy solution. Since we are always picking the assignment with the minimum cost, δ_g is a relatively good result.

Algorithm 1. Greedy algorithm.

```
1. for each parcel p_i do
2.    build a list L_i sorted in ascending order of c(p_i, w_j) for all the workers
3. end for
4. M = {L_1, L_2, ..., L_M}
5. δ_g ← 0
6. while M ≠ ∅ do
7.    (p_i, w_j) ← pick the assignment in M with the minimum cost
8.    if w_j still has capacity to deliver parcel p_i then
9.       assign p_i to w_j
10.      δ_g ← δ_g + cost(p_i, w_j)
11.      M ← M − L_i
12.   else
13.      remove (p_i, w_j) from L_i
14.      continue
15.   end if
16. end while
```

The next step is to estimate the lower bound of any solution $\Delta_{p_i \rightarrow w_j}$ that sets $x(p_i, w_j) = 1$. In our implementation, we set the lower bound of $\Delta_{p_i \rightarrow w_j}$ to be $c(p_i, w_j) + \min_{i' \neq i} c(p_{i'}, w_j)$. In other words, this solution assumes that a worker can take any number of parcels and it always assigns the parcel to the worker with the minimum cost. The proof of the lower bound property is quite straightforward and is omitted in the paper. Given the greedy cost δ_g, we evaluate the lower bound cost of each arc (p_i, w_j) and remove it if the lower bound is higher than δ_g.

Fig. 5 shows an example in which four parcels are assigned to four workers using cost-based pruning. We assume that the capacity of each worker is 2. For each parcel, we build a sorted list in ascending order of the cost. For example, $(w_3, 3)$ in the list of p_1 means the cost of delivering parcel p_1 by worker w_3 is 3. By applying the greedy algorithm, we obtain the assignment results $\{p_1 \rightarrow w_1, p_2 \rightarrow w_3, p_3 \rightarrow w_2, p_4 \rightarrow w_3\}$ with cost 8. Note that the greedy solution in this example is not optimal. The optimal assignment should be $\{p_1 \rightarrow w_1, p_2 \rightarrow w_1, p_3 \rightarrow w_3, p_4 \rightarrow w_3\}$, leading to a cost of 7. Then, we create an array *bound* in which *bound*$[i]$ sums the minimum cost incurred by delivering the parcels except p_i, i.e. *bound*$[i] = \sum_{i' \neq i} \min c(p_{i'}, w_j)$. This array can

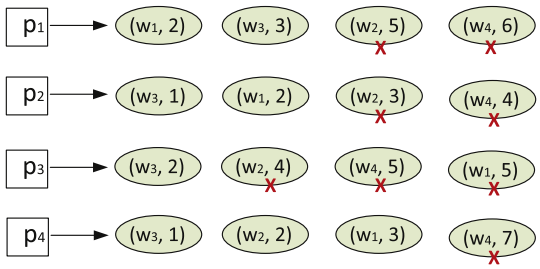

Fig. 5. An example of cost-based pruning.

be pre-computed in advance. In this example, we have $bound[1] = 1 + 2 + 1 = 4, bound[2] = 2 + 2 + 1 = 5$, $bound[3] = 2 + 1 + 1 = 4$ and $bound[4] = 2 + 1 + 2 = 5$. For each entry (p_i, w_j) in list L_i, we calculate the value of $c(p_i, w_j) + bound[i]$ and compare it with the greedy cost 8. All the entries with $c(p_i, w_j) + bound[i] \geqslant 8$ are pruned, as shown in the figure.

5.2. Rule 2: Capacity-based pruning

The cost-based pruning works well when the cost of the greedy solution is very close to optimum. However, since in reality, the worker's capacity is bounded by C, the performance may not be good when C is small. In this part, we propose a new capacity-based pruning strategy, which can guarantee that the number of arcs can be reduced from $P \cdot W$ to $P \cdot \lceil \frac{P}{C} \rceil$. Let $T = \lceil \frac{P}{C} \rceil$ and in reality, T is a value much smaller than W.

Similar to the cost-based pruning, we first build a sorted list L_i for each parcel p_i in ascending order of the delivery cost $c(p_i, w_j)$. The length of each inverted list is W because we can calculate $c(p_i, w_j)$ for all the workers. We use w_{ij} to denote the worker with the j-th lowest cost in L_i to delivery parcel p_i. Intuitively, the workers far away from a parcel's pop-station are unlikely to take the task due to large delivery cost. Theoretically, we can prove that all the workers w_{ij} with $j > T$ can be pruned because the optimal solution will not assign parcel p_i to them.

Lemma 5. *The optimal solution will not assign p_i to the workers w_{ij} with $j > T$.*

Proof. We prove by contradiction. Suppose there exists an optimal solution Δ_{opt} assign which assigns a parcel p_i to a worker w_{ij} with $j > T$ in the sorted list. For the remaining parcels, Δ_{opt} assigns p_m $(m \neq i)$ to worker w_n. Then, we can construct another assignment solution Δ' which follows Δ_{opt} to assign p_m $(m \neq i)$ to worker w_n. At this moment, there are $P - 1$ parcels assigned and we need to determine which worker to deliver p_i. Since the first T workers can take at most $T \cdot C \geqslant P$ parcels, we can find a worker w_{ij} with $j \leqslant T$ who is not fully occupied. Then, Δ' can assign the parcel p_i to this worker, generating a total cost smaller than Δ_{opt}. This contradicts with the assumption that Δ_{opt} is an optimal solution. □

Thus, the capacity-based pruning reduces the number of arcs in the network by $\frac{W \cdot C}{P}$ times. With the availability of a large pool of crowd-workers, the pruning strategy is highly effective.

Its performance can be further improved by avoiding building the sorted lists for all the parcels, whose complexity is $O(P \cdot W \cdot \log W)$. To reduce the CPU time cost in this component, we propose an index-based methodology to avoid calculating the cost $c(p, w)$ for all the pairs of parcels and workers. This is inspired by the observation that even though the parcels to be delivered are dynamic workloads whose destinations can only determined on the fly, we can still estimate the lower bound cost for a worker w to take any parcel located in pop-station s. Since the lower-bound is between a worker and a pop-station, it can be pre-computed in an offline manner and stored as an index. Since the parcels will be assigned to the nearest pop-stations based on the consumer's address, we can guarantee that the parcel must be located in the voronoi cell. For example, suppose a worker has a travel pattern from $A \rightarrow B$, as shown in Fig. 6. The destination of any parcel delivered from pop-station s must be located in the same voronoi cell. Then, we can obtain the lower bound and upper bound cost for a worker to deliver a parcel from a pop-station.

Lemma 6. *When a parcel p at station s is assigned to a worker w with travel pattern $A \rightarrow B$, we have $c(p, w) \geqslant d(A, s) + d(s, B) - d(A, B)$.*

Proof. The proof is quite straightforward based on the triangular inequality: $c(p, w) = d(A, s) + d(s, t) + d(t, B) - d(A, B) \geqslant d(A, s) + d(s, B) - d(A, B)$. □

Lemma 7. *When a parcel p at station s is assigned to a worker w with travel pattern $A \rightarrow B$ and the maximum distance from s to any point in the voronoi cell is r, we have $c(p, w) \leqslant d(A, s) + 2r + d(s, B) - d(A, B)$.*

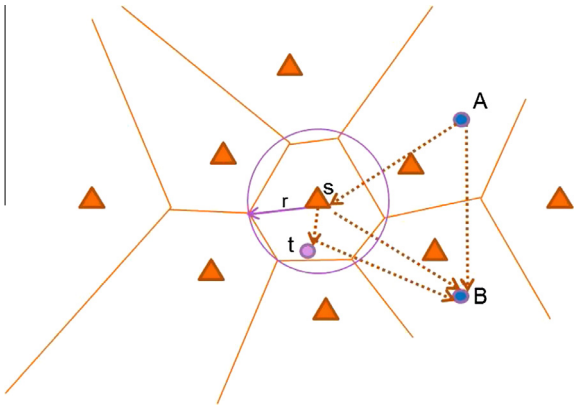

Fig. 6. Lower bound cost for a worker to deliver a parcel in pop-station s.

Proof.

$$\begin{aligned} c(p,w) =& d(A,s) + d(s,t) + d(t,B) - d(A,B) \\ \leqslant& d(A,s) + r + d(t,B) - d(A,B) \\ \leqslant& d(A,s) + r + d(s,t) + t(s,B) - d(A,B) \\ \leqslant& d(A,s) + 2r + t(s,B) - d(A,B) \qquad \square \end{aligned}$$

For the index construction, we build an inverted list with length W for each pop-station, where W is the number of workers. The entries in the list are in the form of a triple $\langle w, LB_w, UB_w \rangle$ and sorted in ascending order of LB_s, where LB_s and UB_s are the lower bound and upper bound derived from Lemmas 6 and 7 respectively. These inverted index are stored in disk and loaded into memory for the optimization of the crowd-delivery model. Given a parcel as an assignment query, our task becomes how to utilize the index to quickly determine the top-T workers. We first determine the pop-station containing the parcel and load the inverted list into memory. Then, we maintain a maximum heap with size T and access the workers in order. For each worker, if the upper bound is smaller than the T-th cost in the heap, denoted by δ_T, we directly insert it into the heap. Otherwise, we calculate the exact cost and compare it with δ_T. If the exact cost is larger than δ_T, we discard the worker. The algorithm terminates when the lower bound of the current worker is larger than δ_k and all the remaining workers can be pruned.

In summary, the computation cost incorporates detecting the top-T workers and running min-cost flow algorithms on the reduced network. The complexity of the former component is $O(P \cdot W \cdot \log T)$ as we use a heap to avoid sorting the whole list. The complexity of the later component is reduced to $O(P^3(P+W))$.

5.3. Rule 3: Frequency-based pruning

Finally, we propose a frequency-based pruning strategy to further reduce the number of arcs in the network. Let W_a be the group of arcs retained after adopting the capacity-based pruning. For the elements in the top-T heap, we sort them again in ascending order of $c(p_i, w_j)$. Our pruning relies on an important observation that if the frequency of a worker w_{ij} in W_a is no greater than C, then all the workers located after w_{ij} in the sorted list L_i can be pruned.

Lemma 8. *Let $freq[w]$ denote the frequency of w occurring in W_a. Given a worker w_{ij} in the i-th sorted list, if $freq[w_{ij}] \leqslant C$, then all the workers $w_{ij'}$ with $j' > j$ can be pruned.*

Proof. Again, we prove by contradiction. Suppose there exists an optimal solution Δ_{opt} assign which assigns a parcel p_i to a worker $w_{ij'}$ with $j' > j$. For the remaining parcels, Δ_{opt} assigns p_m $(m \neq i)$ to worker w_n. Then, we can construct another assignment solution Δ' which follows Δ_{opt} to assign p_m $(m \neq i)$ to worker w_n. Then, we simply need to assign p_i to w_{ij} in Δ' because the worker has enough capacity to take the task. Thus, Δ' is a better solution than Δ_{opt}, which leads to a contradiction. □

Therefore, when we detect such a low-frequency worker in W_a, we can remove a number of workers. After that, we need to decrease the frequencies of these pruned workers by 1, which may lead to further pruning of more workers. The process can iterate until no more workers can be pruned. To make this iterative procedure more efficient, we propose our iterative frequency-based pruning in Algorithm 2. We maintain an array *idx* with size P and initialize the entries in the array to be T. $idx[i]$ stores the smallest index of a worker located in the i-th and with a frequency $freq[w_{ij}] \leqslant C$. The algorithm starts by counting the frequency of each worker in W_a. Then, the process framework consists of two components: (1) scan all the workers and update the *idx* array; and (2) remove workers in the i-th sorted list whose index is larger than $idx[i]$. It terminates when no more workers can be pruned.

Algorithm 2. Iterative frequency-based pruning.

1. build an array *idx* with size P and set $idx[i] \leftarrow T$
2. **while true do**
3. calculate the frequency $freq[w]$ for worker w in W_a
4. $finish \leftarrow$**true**
5. **for** each sorted list L_i for parcel p_i **do**
6. **for** $j = 1; j \leqslant idx[j]; j$++ **do**
7. $w_{ij} \leftarrow L_i[j]$
8. **if** $freq[w_{ij}] \leqslant C$ **then**
9. **if** $j < idx[i]$ **then**
10. $idx[i] \leftarrow j$
11. $finish \leftarrow$**false**
12. **end if**
13. **end if**
14. **end for**
15. **end for**
16. **if** *finish* **then**
17. **break**
18. **end if**
19. **for** $i = 1; i \leqslant P; i$++ **do**
20. keep the first $idx[i]$ workers in list L_i
21. **end for**
22. **end while**

Fig. 7 shows an example of iterative frequency-based pruning. In this example, there are three parcels to be assigned and the T workers have been sorted by $c(p, w)$. We assume that the worker capacity is 2. In the first iteration, we scan the frequencies of all the workers and update the array of *idx*. For example, $idx[3] = 2$ because the frequency of w_8 is no greater than the capacity and all the workers afterwards can be removed. In the second iteration, we can further prune some workers from the lists due to the frequency update. For example, the frequency of w_1 is 3 in the first iteration and becomes 2 after the frequency update. Then, all the workers after w_1 can be pruned. The iteration continues until no more workers can be pruned. In this example, each list for parcel p_i happens to retain only one worker, who will take the delivery task.

6. Experimental study

This section presents results of an extensive performance study of various pruning techniques. All the codes are implemented in C++ and the min-cost flow problem is solved by the network simplex algorithm provided by the API of the LEMON library.[6] We conduct the experiments on a server with 128 GB memory, running Centos 5.6.

6.1. Datasets

We construct three datasets to simulate the experimental environments for the crowd-delivery problem:

- **Singapore bus**. It contains the bus transactions, including boarding and alighting bus stations, of millions of people in Singapore for a period of 6 months. We treat each person as a crowd-worker and detect the most frequent travel routes, say from station A to station B, as the motion pattern. We use the 124 pop-stations of SingPost[7] in our experiments and the destinations of the parcels are generated randomly.
- **Singapore taxi**. It contains trajectories from 13,200 taxies in Singapore over one week. Each taxi continuously reports its locations at a frequency of 20–80 s, which provide us the full historical trajectory. Since the status of the taxi (e.g., busy or vacant) is available, we can easily derive the travel route of each ride. For example, if the status changes from vacant to busy, we know there is a passenger on board. After a while, the status changes back to vacant, we know the destination of the ride. From the taxi dataset, we extract 500,000 most frequent travel routes as the motion patterns of the crowd-workers. The parcels are generated using the SingPost pop-stations and their destinations are randomly assigned.

[6] http://lemon.cs.elte.hu/trac/lemon.
[7] https://www.mypopstation.com/locations.

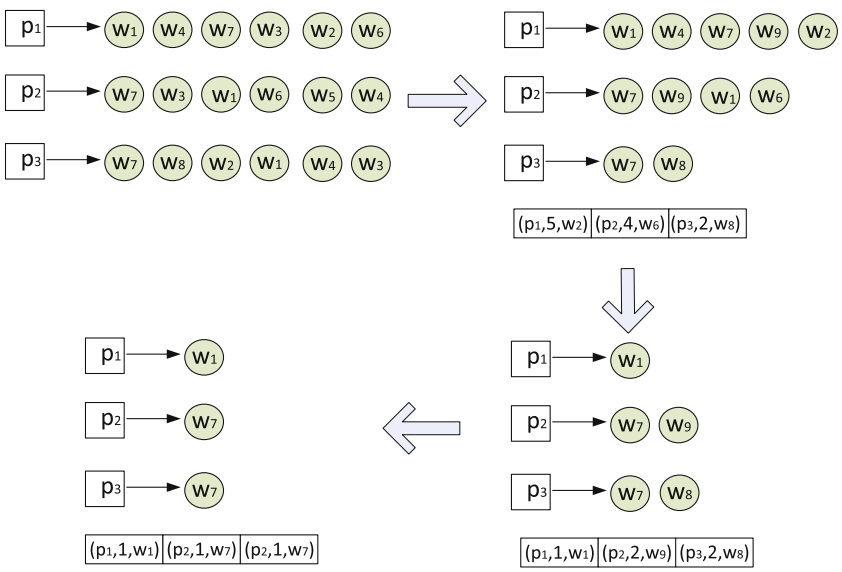

Fig. 7. An example of iterative frequency-based pruning.

- **GeoLife** (Zheng et al., 2009). The dataset essentially keeps all the travel records of 182 users for a period of over three years, including multiple kinds of transportation modes (walking, driving and taking public transportation). For each user, the GPS information is collected periodically and 91 percent of the trajectories are sampled every 15 s. From the dataset, we sample 500,000 sub-trajectories with a period of 30 min as the motion patterns of our crowd-workers. The locations of pop-stations and the destinations of parcels are generated randomly.

In total, we generate 500,000 motion patterns for the crowd-workers and 5000 parcels from each dataset.

6.2. Methods

We report the performance of the following four methods:

- M_0. This method simply reduces the crowd-delivery model to the min-cost flow problem without applying any pruning techniques. It uses as a baseline solution.
- M_1. This method applies the cost-base pruning to reduce the number of arcs in the network.
- M_2. This method applies the capacity-based pruning to further reduce the size of the network generated by M_1.
- M_3. This method applies the frequency-based pruning on the network generated by M_2.

For the performance metric, we report both the number of arcs retained after applying the pruning techniques and the running time of calling the API of the LEMON library to solve the min-cost flow problem in the reduced network. We examine the performance in terms of increasing W (from 100,000 to 500,000), P (from 1000 to 5000) and C (from 1 to 9), with their default values set to $W = 200,000, P = 2000$ and $C = 5$ respectively.

6.3. Performance study

We report the number of arcs between the parcels and workers in Table 2. For method M_0, it does not apply any pruning techniques. Thus, it needs to handle a network with $P \cdot W$ arcs because each parcel can be assigned to any worker. This number could be very huge for large W and P. For instance, when $W = 500,000$ and $P = 2000$, the network contains 1 billion arcs. Method M_1 uses the cost-based pruning to reduce the network size. The method is effective only when C is large or P is small. When $C = 9$, the number of arcs reduces from 400 million to 9 million in the GeoLife dataset. This is because with higher capacity, a parcel is more likely to be assigned to the worker with the minimum delivery cost. Thus, the total cost generated by the greedy method becomes closer to the optimal solution and facilitates better pruning. Similarly, when P is small, the parcels can be assigned to the most suitable workers with less conflict. Hence, when $P = 1000$, it reduces the number of arcs from $200M$ to $3M$ in the Singapore Taxi and GeoLife datasets.

The capacity-based pruning method M_2 is highly effective in reducing the network size. The number of arcs is reduced by orders of magnitude. The method also prefers small P and larger C, but is insensitive to W. This is because it guarantees that the

Table 2
Number of arcs in the network after applying the pruning strategies (result unit = million).

	Increasing W					Increasing P					Increasing C				
	100,000	200,000	300,000	400,000	500,000	1000	2000	3000	4000	5000	1	3	5	7	9
Singapore bus dataset															
M_0	200	400	600	800	1000	200	400	600	800	1000	400	400	400	400	400
M_1	200	368	522	526	555	16	368	600	800	1000	400	400	368	220	111
M_2	0.8	0.8	0.8	0.8	0.8	0.8	0.8	1.8	3.2	5	4	1.33	0.8	0.572	0.446
M_3	0.066	0.035	0.019	0.024	0.025	0.003	0.035	0.256	0.784	2.4	3.8	0.219	0.035	0.008	0.004
Singapore taxi dataset															
M_0	200	400	600	800	1000	200	400	600	800	1000	400	400	400	400	400
M_1	136	115	151	126	94	3	115	476	728	995	391	205	115	47	18
M_2	0.8	0.8	0.8	0.8	0.8	0.2	0.8	1.8	3.2	5	4	1.33	0.8	0.572	0.446
M_3	0.113	0.024	0.011	0.007	0.008	0.003	0.024	0.205	0.606	1	2.6	0.176	0.024	0.005	0.003
GeoLife dataset															
M_0	200	400	600	800	1000	200	400	600	800	1000	400	400	400	400	400
M_1	193	222	160	523	243	3	222	577	779	978	392	383	222	45	9
M_2	0.8	0.8	0.8	0.8	0.8	0.2	0.8	1.8	3.2	5	4	1.33	0.8	0.572	0.446
M_3	0.58	0.461	0.429	0.421	0.394	0.079	0.461	1.3	2.6	4.2	3.7	1	0.461	0.274	0.183

number of arcs will be reduced to P^2/C after applying the pruning, which is a variable irrelevant to W. The frequency-based pruning can further reduce the network size by around an order of magnitude. When C is large, the method works extremely well. For example, after applying the frequency-based pruning, each parcel in the Singapore Taxi dataset has only 2 candidate workers.

The running time of the comparison methods is reported in Table 3. We have the following observations. (1) The implementation of LEMON library is highly efficient. It takes less than 3 h for the baseline solution to find the optimal solution, even in a network with 1 billion arcs. (2) M_1 can significantly reduce the running time when C is large or P is small. For instance, when $P = 1000$, the speed is improved by at least one order of magnitude, and (3) M_2 and M_3 work extremely fast. They can handle most of the scenarios with less than 10 s because the network size has been significantly reduced. In general, M_2 boosts the performance by hundreds of times and the speedup for M_3 can reach three orders of magnitude.

When compared with TRACCS system, our solution based on network min-cost flow model has two major advantages. First, it is very efficient and returns optimal results. The authors of TRACCS system admit that their proposed Integer Linear Programming model is intractable even for small-scale problems. Hence, they can only resort to heuristic methods that return approximate results without any theoretical guarantee. Second, our solution is scalable to handle millions of crowd workers, whereas the performance study of TRACCS system only evaluates up to 1000 agents. As illustrated in the experimental study, the network size reduces as there are more available crowd workers. For a crowd-delivery system to work in practice, it is important that there are a large pool of citizen workers available as the service providers. Consequently, the delivery cost per task can be reduced and the system can be profitable for the logistic companies. Therefore, supporting million-scale crowd workers is a key issue to the success of crowd-delivery systems.

Table 3
Results of running time (result unit = second).

Method	Increasing W					Increasing P					Increasing C				
	100 K	200 K	300 K	400 K	500 K	1000	2000	3000	4000	5000	1	3	5	7	9
Singapore bus dataset															
M_0	1040	2510	3348	4154	5880	1001	2510	3730	5113	8120	1990	2360	2510	2281	2052
M_1	1040	2412	2644	3279	3893	79	2412	3730	5113	8120	1990	2360	2412	1338	778
M_2	5	5	6	7	9	3	5	13	25	38	25	10	5	4	4
M_3	1	2	3	3	5	1	2	3	8	19	23	3	2	2	2
Singapore taxi dataset															
M_0	1105	2014	3016	4533	6112	913	2014	3303	4572	6098	1948	2048	2014	2012	1977
M_1	943	1854	2645	3588	4179	24	1854	2782	4236	6092	1720	1795	1854	1000	316
M_2	6	5	5	6	6	2	5	11	32	41	21	7	5	3	2
M_3	2	2	3	3	4	1	2	3	4	10	12	2	2	2	2
Beijing dataset															
M_0	1373	2339	3580	4397	5677	1008	2339	4499	5926	7542	2214	2561	2339	2071	1998
M_1	1295	1823	1486	3788	1989	20	1823	4214	5899	7523	1910	1823	1127	415	76
M_2	6	6	8	10	10	3	6	21	39	79	35	13	6	5	4
M_3	6	6	7	7	7	2	6	19	36	66	33	12	6	4	3

7. Conclusion and future work

In this paper, we proposed an effective crowd-delivery model by utilizing a large pool of citizen workers to enhance the last-mile delivery. To efficiently solve the model, we formulated it as a network min-cost flow problem and proposed various pruning techniques that can dramatically reduce the network size. The running time was improved by two orders of magnitude from the baseline solution and can support real-time delivery optimization in the large-scale mobile crowd-sourcing problem.

This frontier and novel concept of crowd delivery in city last-mile application has significant impact on the urban logistics development. Compared with traditional last-mile methods or synchronized/consolidated last-mile (where service providers cooperate and share the urban consolidated center to re-assign delivery tasks to vehicles by parcel destination), crowd delivery has the following merits: Firstly, its delivery is in high parallelism and each delivery task is independent. Whereas, the delivery process in the synchronized/consolidated delivery can be considered as sequential. The previous delivery tasks will always impact the subsequent tasks; Secondly, it is one-to-one communication in crowd delivery. However, it is one-to-more communication in traditional delivery; Thirdly, from the environmental and sustainable point of view, crowd sourcing delivery advocates green supply chain and eco-friendly. By using millions of citizens as part-time delivery labor in urban logistics, fewer number of vehicles are needed for service providers, leading to less carbon emission; Last but not least, from the perspective of social revolution and responsibility, this new design in urban logistics will increasingly become one of the most meaningful forms of competitive advantage. As it can cater customers' insatiable appetite for greater simplicity and convenience by adding a layer between service providers and end customers. Crowd delivery offers a way for each individual to participate in social change happening on the ground. This novel design will improve everyone's life.

For Logistics service provider considering implementing crowdsourcing model in the last-mile delivery, our results suggest that crowdsourcing could be applied in real-time delivery requests in the large-scale problem. The proposed effective pruning algorithms make it possible to get optimal solution in an efficient way. However, there are some practical issues and concerns that should be raised and need to be investigated. Data privacy and confidentiality is one of the most key concerns. Parcel package may be redesigned with only barcode and rare limited information. Legal regulations should be enhanced towards governing the collection, use and disclosure of personal data by all service providers and crowd workers. Also general exceptional clause should be emphasized when installing the apps and some penalty rules and obligations should be agreed when accepting the delivery tasks in order to prevent the uncertainties during the delivery. In addition, there are some discussion about the user case. In this paper, the parcel we are referring to normal commerce small items or normal mails. Parcel sensing will reinforced before sending requests to crowd to avoid hazardous and dangerous parcels.

Our work servers as the initial efforts on the problem of crowd-delivery that takes into account the motion patterns of workers and uses the additional travel distance as the cost measurement. Since the system is often much more complex in reality, our future works will take into account (1) the delivery time requirements of consumers; (2) a more practical measure, such as additional travel time, to measure the cost; (3) better pricing strategy and higher system reliability and (4) on-demand requests similar to UberRush, but with a much larger pool of workers.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. Network Flows – Theory, Algorithms and Applications. Prentice Hall.

Benouaret, K., Valliyur-Ramalingam, R., Charoy, F., 2013. Crowdsc: building smart cities with large-scale citizen participation. IEEE Internet Comput. 17 (6), 57–63.

Chen, C., Cheng, S., Gunawan, A., Misra, A., Dasgupta, K., Chander, D., 2014a. TRACCS: a framework for trajectory-aware coordinated urban crowd-sourcing. In: Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2–4, 2014, Pittsburgh, Pennsylvania, USA.

Chen, Z., Fu, R., Zhao, Z., Liu, Z., Xia, L., Chen, L., Cheng, P., Cao, C.C., Tong, Y., Zhang, C.J., 2014b. Gmission: a general spatial crowdsourcing platform. Proc. VLDB Endow. 7 (13), 1629–1632.

Cheng, P., Lian, X., Chen, Z., Fu, R., Chen, L., Han, J., Zhao, J., 2015. Reliable diversity-based spatial crowdsourcing by moving workers. PVLDB 8 (10), 1022–1033.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Manage. Sci. 6, 80–91.

de Souza, R., Goh, M., Lau, H.-C., Ng, W.-S., Tan, P.-S., 2014. Collaborative urban logistics Íc synchronizing the last mile a singapore research perspective. Procedia – Social Behav. Sci. 125, 422–431, Eighth International Conference on City Logistics 17–19 June 2013, Bali, Indonesia.

DellAmico, M., Hadjidimitriou, S., 2012. Innovative logistics model and containers solution for efficient last mile delivery. Procedia – Social Behav. Sci. 48, 1505–1514, Transport Research Arena 2012.

Florian, M., Lebeuf, D., 1997. Network Optimization. Chapter: An Efficient Implementation of the Network Simplex Method. Springer, Berlin, Heidelberg, pp. 265–291.

Handoko, S., Nguyen, D.T., Lau, H.C., 2014. An auction mechanism for the last-mile deliveries via urban consolidation centre. In: 2014 IEEE International Conference on Automation Science and Engineering (CASE), pp. 607–612.

Ilarri, S., Wolfson, O., Delot, T., 2014. Collaborative sensing for urban transportation. IEEE Data Eng. Bull. 37 (4), 3–14.

Kazemi, L., Shahabi, C., 2012. Geocrowd: enabling query answering with spatial crowdsourcing. In: Proceedings of the 20th International Conference on Advances in Geographic Information Systems. SIGSPATIAL '12. ACM, pp. 189–198.

Kennington, J.L., Helgason, R.V., 1980. Algorithms for Network Programming. John Wiley & Sons, Inc., New York, NY, USA.

Király, Z., Kovács, P., 2012. Efficient implementations of minimum-cost flow algorithms. CoRR, abs/1207.6381.

Liakos, P., Delis, A., 2015. An interactive freight-pooling service for efficient last-mile delivery. In: 2015 16th IEEE International Conference on Mobile Data Management (MDM), vol. 2, pp. 23–25.

Petrovic, O., Harnisch, M., Puchleitner, T., 2013. Opportunities of mobile communication systems for applications in last-mile logistics. In: 2013 International Conference on Advanced Logistics and Transport (ICALT), pp. 354–359.

Pournajaf, L., Xiong, L., Sunderam, V.S., Goryczka, S., 2014. Spatial task assignment for crowd sensing with cloaked locations. In: IEEE 15th International Conference on Mobile Data Management, MDM 2014, Brisbane, Australia, July 14–18, 2014, vol. 1, pp. 73–82.
Su, H., Zheng, K., Huang, J., Jeung, H., Chen, L., Zhou, X., 2014. Crowdplanner: a crowd-based route recommendation system. In: ICDE, pp. 1144–1155.
Zhang, C.J., Tong, Y., Chen, L., 2014. Where to: crowd-aided path selection. PVLDB 7 (14), 2005–2016.
Zheng, Y., Zhang, L., Xie, X., Ma, W., 2009. Mining interesting locations and travel sequences from GPS trajectories. In: WWW, pp. 791–800.