8-2021

A Lagrangian column generation approach for the probabilistic crowdsourced logistics planning

Chung-kyun HAN
Singapore Management University, ckhan.2015@phdis.smu.edu.sg

Shih-Fen CHENG
Singapore Management University, sfcheng@smu.edu.sg

2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)
August 23-27, 2021. Lyon, France

A Lagrangian Column Generation Approach for the Probabilistic Crowdsourced Logistics Planning

Chung-Kyun Han[1] and Shih-Fen Cheng[2]

Abstract— **In recent years we have increasingly seen the movement for the retail industry to move their operations online. Along the process, it has created brand new patterns for the fulfillment service, and the logistics service providers serving these retailers have no choice but to adapt. The most challenging issues faced by all logistics service providers are the highly fluctuating demands and the shortening response times. All these challenges imply that maintaining a fixed fleet will either be too costly or insufficient. One potential solution is to tap into the crowdsourced workforce. However, existing industry practices of relying on human planners or worker's self-planning have been shown to be inefficient and laborious. In this paper, we introduce a centralized planning model for the crowdsourced logistics delivery paradigm, considering individual worker's spatio-temporal preferences. Considering worker's spatio-temporal preferences is important for the planner as it could significantly improve crowdsourced worker's productivity. Our major contributions are in the formulation of the problem as a mixed-integer program and the proposal of an efficient algorithm that is based on the column generation and the Lagrangian relaxation frameworks. Such a hybrid approach allows us to overcome the difficulty encountered separately by the classical column generation and Lagrangian relaxation approaches. By using a series of real-world-inspired numerical instances, we have demonstrated the effectiveness of our approach against classical column generation and Lagrangian relaxation approaches, and a decentralized, agent-centric greedy approach. Our proposed hybrid approach is scalable to large problem instances, with reasonable solution quality, and achieves better allocation fairness.**

I. Introduction

For the past two decades, the retail industry has been slowly moving online. During the COVID-19 pandemic, this trend has accelerated tremendously globally, and this shift has put a great strain on the fulfillment infrastructure, particularly the last-mile delivery service. The creation of year-around shopping events by various giant online retailers and the pursuit for ever-shorter delivery time further exacerbate the situation, as the delivery demands are now increasingly spiky and with short turnaround time. This makes it challenging to maintain a traditional logistics team that comprises solely of full-time workers.

To tackle these challenges, many logistics service providers have been adopting the paradigm of crowdsourced (CS) logistics. The CS logistics service could be organized in many shapes and forms, but the general idea is to incorporate part-time workers during the peak period, to make up for the insufficient capacity. Utilizing CS workers has the benefits of not having to commit to the capacities that are only required during demand peaks. However, as more companies are now incorporating this concept, logistics companies are also competing against each other for the pool of CS workers. The new challenge for CS logistics is thus to more efficiently utilize CS workers.

There are two major ways in which firms engage CS workers: 1) the *push-based* approach, where the firm "assign" jobs to CS workers, and 2) the *pull-based* approach, where the CS workers are in charge of selecting jobs. As the push-based approach does not take into account workers' preference, it oftentimes leads to high rejection rate (if the CS platform allows job rejection), low worker satisfaction, or low productivity (the assigned jobs might be inconvenient for CS workers). For the pull-based approach, the issue is the cognitive load for CS workers to schedule the jobs they are interested in taking constantly. Additionally, it is also empirically shown that pull-based scheme often leads to the "super-agent" phenomenon [1], i.e., a small number of dedicated workers tend to dominate the task pool by selecting the most desirable/profitable tasks consistently.

As extensively discussed in [2], due to the uncoordinated and myopic decision making process, CS workers in the pull-based mode perform significantly worse than their peers working in the push-based mode, in terms of both the productivity and the equality of task distribution. However, to effectively implement push-based mode, the central planner would need to take into account individual worker's desired work schedule and their spatiotemporal preference. This is particularly important when a CS worker is truly part-time, and has to be at certain locations (e.g., residence or office) at designated time windows, and only has limited time duration in which he could work on the assigned jobs.

In this research, we extend [2] by defining a task to have both pickup and delivery locations. We also allow planners to specify time windows for all tasks and locations along workers' desired routes. We achieve these additional requirements while considering CS workers' probabilistic desired routes. In summary, we make the following contributions:

- First, we formulate the CS logistics planning problem as a mixed integer linear program (MILP). Our major innovation is the extension of the probabilistic CS planning method to the domain of last-mile logistics, where all tasks now contain both pickup and delivery locations. We also allow the planner to specify the time window requirements for all nodes included in the plan. This is on top of the probabilistic incorporation of CS

[1]Chung-Kyun Han is currently an Engineering Manager with the LG Display ckhan.2015@phdcs.smu.edu.sg

[2]Shih-Fen Cheng is currently an Associate Professor of Computer Science with the School of Computing and Information Systems, Singapore Management University sfcheng@smu.edu.sg

workers' pre-existing routine routes.

- Second, as the MILP formulation is intractable in practice, we develop a novel solution approach that is based on both the column generation (CG) framework and the Lagrangian relaxation (LR) framework. This hybrid approach allows us to address the challenges we encounter when we attempt to implement the classical CG and LR approaches.
- Finally, the performances of all competing approaches are evaluated in the real-world-inspired scenarios based on inputs from our industry partner that relies heavily on the CS workers for their last-mile delivery operations. In our numerical experiments, we quantify both the computational efficiency and the solution quality of our proposed approaches against classical CG and LR approaches as baselines. We also discuss the practical implication of having a central planner versus an agent-centric task selection scheme (the pull-based scheme).

II. Related Literature

Crowdsourcing and Last-Mile Logistics: Before the 2010s, CS tasks are mostly online only (e.g., via Amazon's Mechanical Turk platform), but some authors already suggest the idea of location-specific CS (e.g., see [3]). In recent years, with sensor-rich smartphones becoming increasingly accessible, location-specific CS tasks are becoming more prevalent, and applications such as Uber have demonstrated how CS workers can be effectively incorporated using smartphone Apps. Following this trend, logistics service providers are also seeing CS as a viable operational paradigm (e.g., [4]).

There are many attempts in dealing with a single-point CS task such as environment monitoring and location-specific data collection (e.g., [5], [6]), however, these approaches cannot directly handle last-mile delivery tasks, which have pickup and delivery locations. Some researchers [7] have studied the utilization of CS workers in last-mile logistics, however, they do not consider the pre-existing routine routes of CS workers.

Lagrangian Relaxation and Column Generation: In combinatorial optimization problems, a small group of constraints tends to make the problems complicated. Lagrangian relaxation (LR) [8] and column generation (CG) [9] are well-known optimization techniques for dealing with those complicating constraints. In logistics planning, both approaches are shown to be highly effective (e.g., see [2] and [10])

More recently, researchers also look at different ways of combining CG and LR (e.g., see [11]), which can be classified into two categories: a) using LR on the extended formulation (e.g., [12]), b) using LR on the compact formulation (e.g., [13]). While the The first idea uses LR to approximate dual variables that are inputs for calculating the reduced cost of a new column (decision variable), the second idea generates columns with Lagrangian costs deduced from a compact formulation. Whereas those approaches are more CG-centric, our approach is more LR-centric, which makes ours deal with larger and more realistic problem instances efficiently.

III. Problem Description and Model

As a central planner, we assume that the set of all CS workers (agents) is known to us. All agents are assumed to be part-time and have their own daily routine routes, where each routine route is defined as a sequence of locations that an agent has to visit at designated time windows. These routine routes could be self-reported or inferred from past mobility traces, thus the actual routine route taken is inherently probabilistic. All agents are required to report their capacity limits on volume and weight, and the time budgets they are willing to spend on delivery tasks (their time budgets could be routine-route-dependent). Our objective is to maximize the expected reward from all agents' task fulfillment, subject to agents' capacity limits and their uncertain routine routes. The critical decision for the planner is the set of tasks to assign to each agent, and the suggested delivery sequence depending on an agent's routine route.

Our model is similar to what was proposed by [2], where the task assignment and routing in a CS planning scenario is elegantly captured by a variant of the *team orienteering problem*. To satisfy the requirements of our problem, we introduce the following new enhancements to the model:

- In the previous model, all tasks occur at single locations, while tasks in our model must have both a pickup location and a delivery location, and the pickup location must be visited before the delivery location.
- All locations associated with delivery tasks and agents' routine routes could have time window requirements attached to them.
- Since agents have to physically carry the parcels as they fulfill the delivery tasks, the set of tasks assigned to each agent has to be constrained by the capacity limits. This is in contrast to the previous model, where tasks are performed at the designated location, and no carrying capacity needs to be considered.

A. The Mixed Integer Linear Programming Model

Let A be the set of CS agents and let (v^a, w^a, E^a) be agent a's volume and weight limits, and the set of routine routes respectively. As the routine route choice is assumed to be probabilistic and agent's time budget might depend on which routine route s/he takes, we use tuple (a, e) to denote the instance when agent a follows routine route e, and let $p^{a,e}$ and $u^{a,e}$ be the associated probability and time budget. Let $R^{a,e}$ be the set of nodes belonging to a routine route (a, e), and it should contain at least a pair of origin and destination, denoted as $o^{a,e}$ and $d^{a,e}$. The ordering of nodes in $R^{a,e}$ is determined by $c^{a,e}_{i,j}, i, j \in R^{a,e}$, where $c^{a,e}_{i,j} = 1$ implies that node j immediately follows node i.

Let K be the set of all tasks. Let $(h_k, n_k, v_k, w_k, r_k)$ be task k's pickup location, delivery location, volume, weight, and reward respectively. Note that we allow multiple tasks to share the same pickup location, to reflect the possibility that a warehouse might exist for all agents to pickup tasks. Let $K^{a,e} \subseteq K$ be the set of tasks that can be feasibly completed for route (a, e), and let $P^{a,e} = \{h_k \mid k \in K^{a,e}\}$

and $D^{a,e} = \{n_k \mid k \in K^{a,e}\}$ be the set of all pickup and delivery locations belonging to $K^{a,e}$. For simplicity, we denote all nodes reachable by (a, e) as $N^{a,e}$, and $N^{a,e} = P^{a,e} \cup D^{a,e} \cup R^{a,e}$.

Our problem is defined on a complete directed graph, which is the union of all (a, e) tuples. Formally speaking, this complete graph is defined as $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \bigcup_{a \in A, e \in E^a} N^{a,e}$ and $\mathcal{A} = \mathcal{N} \times \mathcal{N}$. For each node i, we specify its time window as $[\alpha_i, \beta_i]$, and we use $t_{i,j}$ to be the sum of the travel time from i to j and the service time at node i (e.g., when an agent carries out the task at i).

There are four sets of decision variables in our mixed integer linear programming (MILP) model:

- $y_k^a \in \{0, 1\}$: set to 1 if task k is assigned to agent a.
- $z_k^{a,e} \in \{0, 1\}$: set to 1 if task k is assigned to agent a, yet cannot be completed under the routine route e.
- $x_{i,j}^{a,e} \in \{0, 1\}$: set to 1 when agent a moves from i to j under the routine route e.
- $\mu_i^{a,e} \in \mathbb{R}_{\geq 0}$: the time when agent a visits node i under routine route e.

The objective of the MILP model is to maximize the sum of the expected rewards from all agents. Since the probability that an assigned task k to agent a cannot be finished is $(\sum_{e \in E^a} p^{a,e} z_k^{a,e})$, the objective function is defined as:

$$\max_{y_k^a, z_k^{a,e}, x_{i,j}^{a,e}, \mu_i^{a,e}} \sum_{k \in K} r_k \sum_{a \in A} \left(y_k^a - \sum_{e \in E^a} p^{a,e} z_k^{a,e} \right). \quad (1)$$

The first group of constraints handles the task assignment. In (2), we require that each task is assigned to at most one agent. In (3), we state that an agent is only assigned tasks they can feasibly complete. An agent's volume and weight capacity limits are enforced in (4) and (5). In (6), we ensure the logic that $z_k^{a,e}$ cannot be 1 unless y_k^a is set to 1.

$$\sum_{a \in A} y_k^a \leq 1, \quad \forall k \in K, \quad (2)$$

$$y_k^a = 0, \quad \forall a \in A, k \in K \setminus K^{a,e}, \quad (3)$$

$$\sum_{k \in K} v_k y_k^a \leq v^a, \quad \forall a \in A, \quad (4)$$

$$\sum_{k \in K} w_k y_k^a \leq w^a, \quad \forall a \in A, \quad (5)$$

$$z_k^{a,e} \leq y_k^a, \quad \forall a \in A,\ \forall e \in E(a),\ \forall k \in K. \quad (6)$$

The second group of constraints deals with routing plans for each (a, e): we specify in (7) and (8) that agent a must depart from $o^{a,e}$, end in $d^{a,e}$, and pass through all nodes in the routine route e. In (9), we enforce that a task can only be delivered if it is picked up. Finally, (10) enforces flow conservation at all pickup and delivery nodes.

$$\sum_{j \in N^{a,e}} x_{o^{a,e},j}^{a,e} = \sum_{j \in N^{a,e}} x_{j,d^{a,e}}^{a,e} = 1, \quad (7)$$

$$\sum_{i \in N^{a,e}, i \neq j} x_{i,j}^{a,e} = \sum_{i \in N^{a,e}, i \neq j} x_{j,i}^{a,e} = 1, \forall j \in R^{a,e} \setminus \{o^{a,e}, d^{a,e}\}, \quad (8)$$

$$\sum_{j \in N^{a,e}} x_{n_k,j}^{a,e} \leq \sum_{j \in N^{a,e}} x_{j,h_k}^{a,e}, \quad \forall k \in K^{a,e}, \quad (9)$$

$$\sum_{j \in N^{a,e}} x_{i,j}^{a,e} = \sum_{j \in N^{a,e}} x_{j,i}^{a,e} \leq 1, \quad \forall i \in P^{a,e} \cup D^{a,e}. \quad (10)$$

The third group of constraints deals with the time window requirements for each tuple (a, e). The arrival time at the origin node is specified in (11). In (12), we relate the arrival times for any pair of nodes i and j based on the routing variable $x_{i,j}^{a,e}$. Note that M represent a large enough constant such that when $x_{i,j}^{a,e} = 0$, this constraint could become non-binding. Time window requirements at all nodes are enforced by (13). In (14), we translate the precedence relationships (captured by $c_{i,j}^{a,e}$) into arrival times ($\mu_i^{a,e}$ and $\mu_j^{a,e}$). Finally, (15) ensures that agent a would visit the pickup node h_k before the delivery node n_k.

$$\mu_{o^{a,e}}^{a,e} = \alpha_{o^{a,e}}, \quad (11)$$

$$\mu_i^{a,e} + t_{i,j} \leq \mu_j^{a,e} + M \cdot (1 - x_{i,j}^{a,e}), \quad \forall i, j \in N^{a,e}; i \neq d^{a,e}, j \neq o^{a,e}, \quad (12)$$

$$\alpha_i \leq \mu_i^{a,e} \leq \beta_i, \quad \forall i \in N^{a,e}, \quad (13)$$

$$c_{i,j}^{a,e} \mu_i^{a,e} \leq \mu_j^{a,e}, \quad \forall i, j \in R^{a,e}, \quad (14)$$

$$\mu_{h_k}^{a,e} \leq \mu_{n_k}^{a,e} + M(1 - \sum_{j \in N^{a,e}} x_{n_k,j}^{a,e}), \quad \forall k \in K^{a,e}. \quad (15)$$

The time budget for each (a, e) is constrained by (16).

$$\sum_{i,j \in N^{a,e}; i \neq d^{a,e}, j \neq o^{a,e}} t_{i,j} x_{i,j}^{a,e} \leq u^{a,e}. \quad (16)$$

Finally, we must ensure that the task assignment and the route planning are consistent:

$$y_k^a - \sum_{j \in N^{a,e}} x_{j,n_k}^{a,e} \leq z_k^{a,e}, \forall a \in A, e \in E(a), k \in K^{a,e}. \quad (17)$$

IV. A Lagrangian Column Generation Approach

Our mathematical formulation can potentially be solved by using the LR approach [2] or the CR approach [14], however, due to the additional constraints we introduce, neither approach works well in its current form. Summarizing the experience of past implementation of these heuristic approaches, we note that for the LR approach, we cannot execute the primal extraction efficiently; while for the CG approach, we cannot efficiently compute reduced costs and generate columns. On the other hand, when focusing on the strengths of these two methods, we see that the master problem of the CG approach, which decides what columns to choose, can be solved in a straightforward and efficient manner; for the LR approach, its strength lies in the handling of subproblems, which is intuitive and efficient.

To take advantage of the strengths of the two approaches, we define paths as columns and follow the CG framework to maintain the primal solution. The primal solution obtained in the CG framework is sent to the LR framework, which would determine the update of the Lagrangian multipliers, and thus affecting the solving of subproblems. As the columns are now generated based on Lagrangian costs instead of reduced costs, we formally call our approach the Column Generation with Lagrangian Costs (CG-LC). For the remainder of the section, we would formally introduce the CG and the LR components in our approach. The effectiveness of our CG-LC approach against classical CG and LC approaches is assessed in the numerical study section.

A. The Column Generation Component

In our CG design, we define feasible paths (those paths that could be feasibly traversed by an (a, e) pair) as columns, and we use set Ω to represent all such feasible paths. We denote $\Omega' \subset \Omega$ as the current column set, and it is initialized to include paths covering only routine routes for all (a, e) tuples (i.e., no task would be performed initially). The objective function based on current Ω' is defined as:

$$\max \sum_{a\in A, e\in E^a} p^{a,e} \sum_{\omega\in\Omega'(a,e)} \theta_\omega \sum_{k\in K(\omega)} r_k, \tag{18}$$

where $\Omega'(a, e)$ refers to paths that belong to the tuple (a, e) in Ω'. We define θ_ω as the binary decision variable, indicating whether a path ω should be included in the solution. By pre-computing $f_\omega = p^{a,e} \sum_{k\in K(\omega)} r_k$, we could avoid duplicative computation and simplify (18) as:

$$\max \sum_{\omega\in\Omega} f_\omega \theta_\omega. \tag{19}$$

The selection of paths is constrained by:

$$\sum_{\omega\in\Omega'(a,e)} \theta_\omega = 1, \quad \forall a \in A,\ \forall e \in E^a, \tag{20}$$

$$e^\omega_k \theta_\omega \le y^a_k, \quad \forall a \in A,\ \forall \omega \in \Omega'(a),\ \forall k \in K, \tag{21}$$

where (20) enforces that exactly one path should be selected for each (a, e) pair, and (21) extracts the binary task assignment decision variable y^a_k from path selection θ_ω. The value y^a_k is set to 1 if the selected path ω includes task k for agent a. For each path ω, we use the binary parameter e^ω_k to indicate whether task k is included in ω.

The assignment of tasks (y^a_k) is further subjected to the following constraints:

$$\sum_{a\in A} y^a_k \le 1, \quad \forall k \in K, \tag{22}$$

$$y^a_k = 0, \quad \forall a \in A,\ \forall e \in E^a,\ \forall k \in K \setminus K^{a,e}, \tag{23}$$

where (22) ensures that each task k is assigned to at most one agent, and (23) ensures that agents would only be assigned feasible tasks (reachable from at least one routine route).

The above formulation is commonly called the *restricted master problem* in the CG literature. From this formulation, we guarantee that we would assign exactly one feasible path to each (a, e) tuple, and as the content of the set Ω' is increasing monotonically, we also ensure that the primal objective value would increase monotonically. This feature is helpful for the LR component as a steadily improving primal value would help to stabilize the LR process.

B. The Lagrangian Relaxation Component

In our MILP formulation, (17) is the set of coupling constraints that connects task assignment to agents to the actual routing given different (a, e) tuple. By dualizing (17), we can effectively decompose the grand MILP model into one assignment subproblem and $\sum_{a\in A} |E^a|$ routing subproblems. The dualized objective function is:

$$L(\boldsymbol{\lambda}) = - \min_{y^a_k, z^{a,e}_k, x^{a,e}_{i,j}, \mu^{a,e}_i} \left(\sum_{k\in K} r_k \sum_{a\in A} \left(\sum_{e\in E^a} p^{a,e} z^{a,e}_k - y^a_k \right) + \sum_{a\in A} \sum_{e\in E^a} \sum_{k\in K^{a,e}} \lambda^{a,e}_k \left(y^a_k - z^{a,e}_k - \sum_{j\in N^{a,e}} x^{a,e}_{j,n_k} \right) \right). \tag{24}$$

Following the LR literature, our dual problem is defined to minimize the negative primal objective function (corresponding to the maximization in the primal); to make the primal and the dual values comparable, we insert another minus sign in front. For each constraint (a, e, k) from (17), we associate it with a positive Lagrangian multiplier $\lambda^{a,e}_k$, and this could be viewed as a penalty when the constraint is violated.

1) Subproblems: Define the *assignment subproblem* by collecting terms and constraints associated with y^a_k and $z^{a,e}_k$:

$$L^1(\boldsymbol{\lambda}) = - \min_{y^a_k, z^{a,e}_k} \left(\sum_{k\in K} r_k \sum_{a\in A} \left(\sum_{e\in E^a} p^{a,e} z^{a,e}_k - y^a_k \right) + \sum_{a\in A} \sum_{e\in E^a} \sum_{k\in K^{a,e}} \lambda^{a,e}_k \left(y^a_k - z^{a,e}_k \right) \right), \tag{25}$$

together with constraints (2) – (6). This subproblem can be solved exactly using a standard solver. Similarly, define the *routing subproblems* by collecting terms and constraints associated with $x^{a,e}_{i,j}$ and $\mu^{a,e}_i$ for each (a, e):

$$\begin{aligned} L^2(a, e, \boldsymbol{\lambda}) &= - \min_{x^{a,e}_{i,j}, \mu^{a,e}_i} - \sum_{k\in K^{a,e}} \lambda^{a,e}_{k,t} \sum_{j\in N(a,e)} x^{a,e}_{j,n_k} \\ &= \max_{x^{a,e}_{i,j}, \mu^{a,e}_i} \sum_{k\in K^{a,e}} \lambda^{a,e}_k \sum_{j\in N^{a,e}} x^{a,e}_{j,n_k} \quad , \end{aligned} \tag{26}$$

together with constraints (7) – (15). For each (a, e), the above formulation is essentially an orienteering problem with a pre-existing routine route, and is known to be NP-hard. Although exact solutions can be obtained for smaller instances, for larger instances we can only approximate the solutions. In our implementation, we use both a branch-and-cut approach (if time allows) and a greedy-based heuristic, which iteratively inserts a feasible task with highest reward into the current route. The routing subproblem solver is modularized, and we could easily replace it if a better solver is identified.

2) Updating Lagrangian Multipliers: The intuition behind the LR approach is to penalize the violation of dualized constraints in the objective function using the Lagrangian multipliers ($\boldsymbol{\lambda}$). The more severe the violation is (in terms of the magnitude of the constraint violation), the higher the multiplier value should be.

The values of the Lagrangian multipliers are updated after all the subproblems are solved using the current Lagrangian multipliers ($\boldsymbol{\lambda}_t$, where t in the current iteration). We follow the standard update procedure as suggested by [8]:

$$\lambda^{a,e}_{k,t+1} \leftarrow \max \left\{ 0, \lambda^{a,e}_{k,t} + a_t \left(y^a_k - z^{a,e}_k - \sum_{j\in N^{a,e}} x^{a,e}_{j,n_k} \right) \right\},$$

where a_t is the step size, and is defined as:

$$a_t = \frac{u_t (L(\boldsymbol{\lambda}_t) - P^*)}{\sum_{a\in A} \sum_{e\in E^a} \sum_{k\in K} \left(y^a_k - z^{a,e}_k - \sum_{j\in N^{a,e}} x^{a,e}_{j,n_k} \right)^2}.$$

In the above formula, P^* refers to the latest primal value from the CG component, and u_t is a parameter initially set to 2 and we would decrease its value by half if there has been no improvement on $L(\boldsymbol{\lambda}_t)$ for three consecutive iterations. The parameter u_t helps to stabilize Lagrangian dynamics.

As noted earlier, there is no need to extract the primal solution, since the CG component has taken care of it.

3) Generating Columns: At the end of the current LR step, we could extract new columns and their associated parameters from the routing subproblem solutions. For each (a, e) tuple, a corresponding path ω' is constructed using $x_{i,j}^{a,e}$, and we append it to Ω'. We should also identify $K(\omega')$ and $f(\omega')$, which are the collection of visited tasks and collected rewards by the path ω'. We also set $e_k^{\omega'} = 1$, if $k \in K(\omega')$, and 0 otherwise.

V. Numerical Experiments

In this section, we evaluate the performance and efficiency of our proposed CG-LC approach against classical CG and LR approaches and a greedy heuristic that mimics a pull-based task selection scheme where individual agents greedily select tasks asynchronously. The numerical instances we use are generated using real-world data from various open sources in Singapore. All approaches are implemented in C++ and tested in an identical hardware/software environment (a server with 72 CPU cores and 512GB RAM, running Red Hat Linux v6.9). The LP and MILP models are solved by IBM CPLEX 12.8.

A. Problem Instance Generation

Our problem instances are composed of the following four categories of data: 1) the pickup locations of tasks (warehouses or depots), 2) the delivery locations of tasks, 3) the routine routes of CS agents, and 4) the travel time between any pair of locations.

To make our instances real-world-like, we use a number of data from Singapore. We generate pickup locations from a small set of actual warehouse locations. The delivery locations are from based on the parcel locker locations provided by two major providers Singapore Postal Service (https://www.speedpost.com.sg/locate-us) and Ninja Van (https://www.ninjavan.co/en-sg/ninja-points). The CS worker's routine routes are derived from a public transit dataset, which captures the origins and destinations of millions of trips. Finally, the travel times between locations are estimated using a map service offered by the Singapore government (https://www.onemap.sg/)

The sizes of the problem instances are based on the number of agents and tasks, denoted as (na, nt). Given (na, nt), agents and tasks are sampled from the datasets. The agent's volume and weight capacity limits are fixed at 5. The volume and weight of a task are generated uniformly between (0, 1], while the reward equals $\max$(volume, weight). We assume that an agent is willing to spend 5% of his usual commute time doing deliveries. The number of routine routes for each agent is either 2 or 3. For each (na, nt) scenario, we generate 20 random instances.

B. Baselines and Solution Approaches

All evaluated approaches are listed in Table I. For each approach, the "router" column indicates how the routing subproblem is solved, while the "coordinator" column indicates how conflicts from multiple agents are coordinated. The *PureGH* is an asynchronous greedy heuristic that mimics human agent's decision making approach, i.e., picking the best available tasks around when s/he is free. The *CGRC* is the classical CG that uses *reduced costs* to generate new columns, where task bundles are columns.

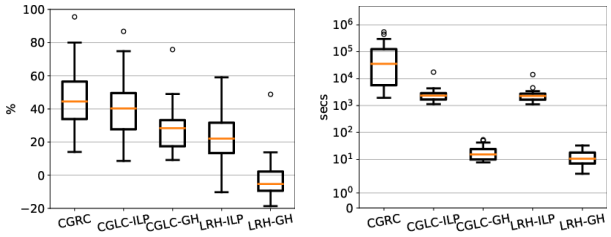
TABLE I: All solution approaches to be compared.

Name	Coordinator	Router
PureGH	-	-
CGRC	CG w/ Reduced Costs	-
LRH-ILP	LR Heuristic	Exact
LRH-GH	LR Heuristic	Greedy Heuristic
CGLC-ILP	CG w/ Lagrangian Costs	Exact
CGLC-GH	CG w/ Lagrangian Costs	Greedy Heuristic

C. Impact of the Decomposition Schemes

We first look at an instance with moderate size of (na, nt) = (10, 40), where the results are summarized as two box plots in Fig. 1. In Fig. 1(a), we compute the relative solution quality of all approaches against the PureGH approach in percentage. The boxes visualize the variability of the results, where the box covers the first (Q1) to the third (Q3) quartiles. The line in the box shows the median and the whisker extends to 1.5(Q3-Q1) above and below; any points that are outside this range are plotted as outliers.

The execution speeds are summarized in Fig. 1(b). The PureGH approach is not included as it finishes almost instantly. For the rest of the approaches, their execution times are plots along a log-scale y-axis using boxes. From this comparison, we can see that CGLC-GH and LRH-GH are equally fast, while CGLC-ILP and LRH-ILP are the second band (around 100 times slower), and CGRC is the slowest (more than 1000 times slower). From this comparison, we can see that the choice of the routing algorithm seems to be most critical in the overall execution speed. It is worth noting that although CGLC-GH and LRH-GH are equal in execution speed, CGLC-GH is significantly better than LRH-GH in solution quality.



(a) Relative solution quality. (b) CPU time.

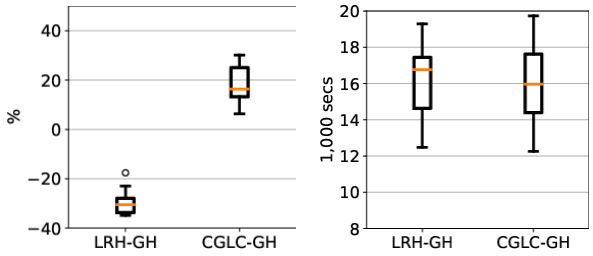
Fig. 1: Instances (na, nt) = (10, 40).

D. Scalability and Fairness

This last subsection evaluates the scalability and performance of LRH-GH and CGLC-GH against PureGH for larger instances. We only include GH-based approaches since all other approaches (CGRC, LRH-ILP, CGLC-ILP) cannot scale to this size.

We summarize our findings in Fig. 2 for the instance of (na, nt) = (40, 960). Consistent with our experience with

smaller instances, although the execution speeds of LRH-GH and CGLC-GH are roughly the same, the solution quality of CGLC-GH is far better than that of LRH-GH: while LRH-GH is worse than PureGH by around 30%, CGLC-GH is better than PureGH by almost 20%.



(a) Relative solution quality. (b) CPU time.

Fig. 2: Instances (na, nt) = (40, 960).

Lastly, as the solver is to be used in the CS setting, we would want to examine the task assignment fairness among workers. In Fig. 3 we plot the standard deviation of obtained rewards among all agents for all (40, 960) problem instances. The boxplot shows the distribution of the standard deviation for each solution approach. Here, whereas PureGH is a decentralized approach, the other two approaches centrally coordinate agents by LRH or CGLC, which result in lower standard deviations among agent's expected rewards (however, we do notice that the distribution of measured standard deviations is highly variable). In other words, central coordination is by design more effective in enabling a fairer task assignment (although fairness is not directly incorporated in the objective of the model).

To conclude, as a heuristic for planning CS logistics operations, our CGLC-GH approach provides an ideal balance among solution quality, computation time, and allocation fairness.

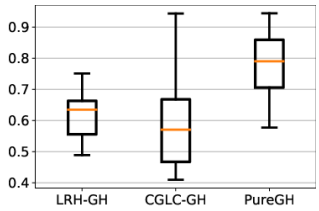


Fig. 3: The reward fairness between agents for each solution approach in (na, nt) = (40, 960) instances.

VI. Conclusions and Future Work

In this paper, we formally define the crowdsourced logistic planning problem as a mixed-integer program. To solve this problem, we introduce a novel hybrid heuristic approach that combines the strengths of both the column generation and the Lagrangian relaxation approaches. While the primal solutions are obtained by running the column generation framework, the actual generation of columns is achieved by the Lagrangian relaxation approach. We use a series of real-world-inspired numerical instances to demonstrate the effectiveness of our approach. By comparing our approach against classical column generation and Lagrangian relaxation approaches, and a decentralized, agent-centric greedy approach, we demonstrate that our proposed hybrid approach is scalable to large problem instance, with reasonable solution quality, and achieves better allocation fairness.

Based on our observations on the computational results, we see that the routing subproblem dominates the computational time and directly impact the solution quality. As such, improving how we solve routing subproblems will be most promising in further enhancing the real-world performance of our approach.

Finally, we think that the idea of combining column generation and Lagrangian relaxation is promising not just for our problem, but potentially to a wider classes of planning problems. We are particularly interested in understanding more about the theoretical properties of such design and the computational performance under different setups.

References

[1] M. Musthag and D. Ganesan, "The role of super agents in mobile crowdsourcing," in *4th Human Computation Workshop*, 2012.

[2] S.-F. Cheng, C. Chen, T. Kandappu, H. C. Lau, A. Misra, N. Jaiman, R. Tandriansyah, and D. Koh, "Scalable urban mobile crowdsourcing: Handling uncertainty in worker movement," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 3, pp. 26:1–26:24, 2018.

[3] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," in *World Sensor Web Workshop*, 2006.

[4] X. Guo, Y. J. L. Jaramillo, J. Bloemhof-Ruwaard, and G. Claassen, "On integrating crowdsourced delivery in last-mile logistics: A simulation study to quantify its feasibility," *Journal of Cleaner Production*, vol. 241, p. 118365, 2019.

[5] C. Chen, S.-F. Cheng, H. C. Lau, and A. Misra, "Towards City-scale Mobile Crowdsourcing: Task Recommendations under Trajectory Uncertainties," *Proceedings of the 24th International Conference on Artificial Intelligence*, pp. 1113–1119, 2015.

[6] L. Tran, H. To, L. Fan, and C. Shahabi, "A Real-Time Framework for Task Assignment in Hyperlocal Spatial Crowdsourcing," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 3, pp. 37:1–37:26, 2018.

[7] A. M. Arslan, N. Agatz, L. Kroon, and R. Zuidwijk, "Crowdsourced Delivery—A Dynamic Pickup and Delivery Problem with Ad Hoc Drivers," *Transportation Science*, vol. 53, no. 1, pp. 222–235, 2019.

[8] M. L. Fisher, "The Lagrangian relaxation method for solving integer programming problems," *Management Science*, vol. 50, no. 12, pp. 1861–1871, 2004.

[9] G. Desaulniers, J. Desrosiers, and M. M. Solomon, Eds. Boston, MA: Springer US, 2005.

[10] J.-Q. Li, P. B. Mirchandani, and D. Borenstein, "Real-time vehicle rerouting problems with time windows," *European Journal of Operational Research*, vol. 194, pp. 711–727, 2009.

[11] D. Huisman, R. Jans, M. Peeters, and A. P. Wagelmans, *Combining Column Generation and Lagrangian Relaxation*. Boston, MA: Springer US, 2005, pp. 247–270.

[12] A. Löbel, "Vehicle Scheduling in Public Transit and Lagrangean Pricing," *Management Science*, vol. 44, no. 12-part-1, pp. 1637–1649, 1998.

[13] D. Huisman, R. Freling, and A. P. M. Wagelmans, "Multiple-Depot Integrated Vehicle and Crew Scheduling," *Transportation Science*, vol. 39, no. 4, pp. 491–502, 2005.

[14] R. El-Hajj, A. Moukrim, B. Chebaro, and M. Kobeissi, "A column generation algorithm for the team orienteering problem with time windows." *Proceedings of the 45th International Conference on Computers & Industrial Engineering*, 2015.