Efficient and Effective Express via Contextual Cooperative Reinforcement Learning

Yexin Li [1], Yu Zheng [2], Qiang Yang [1]
[1] Hong Kong University of Science and Technology, Hong Kong
[2] JD Intelligent Cities Business Unit, Beijing, China
yliby@connect.ust.hk, msyuzheng@outlook.com, qyang@cse.ust.hk

ABSTRACT

Express systems are widely deployed in many major cities. Couriers in an express system load parcels at transit station and deliver them to customers. Meanwhile, they also try to serve the pick-up requests which come stochastically in real time during the delivery process. Having brought much convenience and promoted the development of e-commerce, express systems face challenges on courier management to complete the massive number of tasks per day. Considering this problem, we propose a reinforcement learning based framework to learn a courier management policy. Firstly, we divide the city into independent regions, in each of which a constant number of couriers deliver parcels and serve requests cooperatively. Secondly, we propose a soft-label clustering algorithm named Balanced Delivery-Service Burden (BDSB) to dispatch parcels to couriers in each region. BDSB guarantees that each courier has almost even delivery and expected request-service burden when departing from transit station, giving a reasonable initialization for online management later. As pick-up requests come in real time, a Contextual Cooperative Reinforcement Learning (CCRL) model is proposed to guide where should each courier deliver and serve in each short period. Being formulated in a multi-agent way, CCRL focuses on the cooperation among couriers while also considering the system context. Experiments on real-world data from Beijing are conducted to confirm the outperformance of our model.

CCS CONCEPTS

• **Information systems → Spatio-temporal systems**

KEYWORDS

Express system, Constrained Clustering, Reinforcement Learning

ACM Reference format :

Yexin Li, Yu Zheng, Qiang Yang. 2019. Efficient and Effective Express via Contextual Cooperative Reinforcement Learning. In The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4-8, 2019, Anchorage, AK, USA, 10 pages. DOI: https://doi.org/10.1145/3292500.3330968

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330968

1. INTRODUCTION

Express systems are widely deployed in many major cities, providing much convenience and promoting the development of e-commerce. Fig. 1 A) shows how couriers in an express system work. A courier c_w loads some parcels at the transit station s_v and then delivers them, e.g. d_1 and d_2, to their destinations one by one via a small delivery van. Meanwhile, there may come some pick-up requests from end users during the delivery process, e.g. r_1, each of which is associated with a service location; c_w also tries to go to these places to serve them when delivering. Specially, a pick-up request should be served in a short waiting time, e.g. one hour, while the duration for a parcel to be delivered to its customer can be longer, e.g. several hours or even one day, as it already spent days before the parcel arrived at its transit station. Couriers are required to depart from and return to transit station by specific time [4], to fit the schedule of trucks which send and pick packages to or from stations regularly. We name this specific time interval as an **episode**.

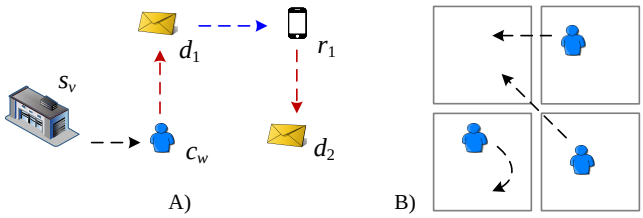


Fig. 1. An express system

Currently, operators try to complete the massive number of tasks by hiring more couriers. However, according to historical data, 70 percent of them complete no more than 20 tasks per day, leading to low revenue and labor waste. Besides, immoderate recruitment cannot solve the efficiency issue in system operation fundamentally. In this paper, we try to address this problem from another perspective, i.e. given a portion of the currently hired couriers, we intelligently manage them to work effectively and cooperatively, thus to complete most tasks in each episode.

Courier management in each episode includes two steps, i.e. how to dispatch parcels to each courier at transit station; how to determine where each courier should deliver and serve in each short period. Fig. 1 B) gives an example, where the city is partitioned into uniform grids. At the beginning of each period t, each courier c_w determines which surrounding grid to go or just

stay at the current one; afterwards, c_w delivers parcels and serves requests only in the selected grid during t until the next period $t+1$, when all couriers choose their next grids again. As we can see, there are nine actions to select from for each courier at each period, i.e. $\{1, 2, \dots, 8\}$ for the eight surrounding grids and 0 for the current one. By these two steps, we want to maximize the total number of completed tasks in the episode. Challenges of this problem can be summarized into three-fold as follows.

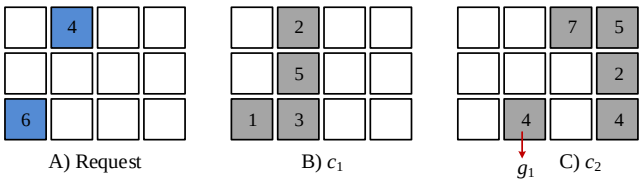


Fig. 2. Parcel allocation at transit station

Urban express systems are large and dynamic. According to historical express data in Beijing, there are tens of thousands of parcels to be delivered every day, so are the pick-up requests. To complete these tasks, hundreds or even thousands of couriers are needed. How to manage this large number of couriers to work cooperatively is important but nontrivial. Besides, pick-up requests come stochastically in real time during the delivery process, making the systems very dynamic.

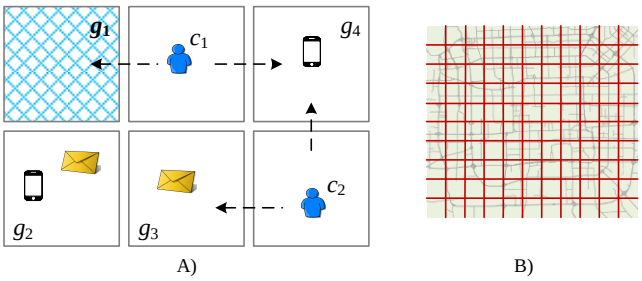


Fig. 3. Online delivery-service process

Being impacted by multiple factors, parcel allocation to couriers at transit station is nontrivial. Firstly, expected pick-up requests that may come later in the episode should be considered as they impact the work burden of each courier. Fig. 2 gives an example; A) shows the hot grids of requests, where a grid with a number means how many requests are expected to be located there in the future; B) shows the parcel allocation to courier c_1 where a grid with a number means how many parcels c_1 has to deliver there. As we can see, c_1 gets 11 parcels; as her delivery grids are very possible to have about 10 requests later, her delivery-service burden is 11 + 10 = 21. Secondly, it is better that all couriers have similar work burdens when departing from transit station, thus to ensure almost even revenues, reduce labor waste and avoid overwork. Fig. 2 C) shows the parcel allocation to c_2, whose delivery-service burden is 22 parcels without any expected request in her delivery grids, thus is almost the same with that of c_1. Lastly, we want the parcels allocated to each courier have destinations in neighboring grids, e.g. the parcel allocation to c_2 is not good as c_2 cannot arrive at grid g_1 from any of her other delivery grids without visiting a third one, where she has no parcel to deliver nor expected request to serve.

Online delivery-service process is too complicated to optimize for a long time. After loading parcels and departing from transit station, each courier chooses where to go and work at the beginning of each period. As shown in Fig. 3 A), each action of a courier has long-term influence, e.g. if courier c_2 goes to g_4 instead of g_3, she can complete the same number of tasks in the current period, i.e. 1 service instead of 1 delivery; however, in the next period, she cannot arrive at g_2 for the two tasks; therefore, we want to optimize a sequence of actions thus to maximize the total number of completed tasks in a long time. Even for a single courier, optimization cannot handle the large solution space, i.e. 9^T where T is the number of periods in the episode. Besides, there are random factors in system operation, e.g. stochastically coming requests and noise, which compromise the accuracy of optimization model. Cooperation among couriers should also be considered, e.g. c_1 and c_2 cannot both go to grid g_4 for the 1 request. Furthermore, context of the system is another important constraint, e.g. c_1 is not suggested to go to g_1 if she does not have any parcel to deliver there nor g_1 is not expected to have any pick-up request later.

Our contributions in this paper are four-fold as follows.

- By Connected Component Detection, we partition the entire city into independent regions and respectively focus on each of them, to reduce problem complexity.
- A soft-label clustering algorithm named Balanced Delivery-Service Burden (BDSB) is proposed to allocate parcels to each courier at transit station.
- A Contextual Cooperative Reinforcement Learning (CCRL) model is proposed to guide where should each courier go at the beginning of each period. Being formulated in a multi-agent way, CCRL tries to guarantee the cooperation among couriers. Besides, system context is considered by pruning.
- Experiments on real-world data from Beijing are conducted to confirm the outperformance of our model.

2. OVERVIEW

Some notations and terminologies used in our paper are defined in this section; then we overview the framework.

Tab. 1. Notations

s_w	A transit station in the system
g_w	A grid in the city
c_w	A courier in the system
d_w / r_w	A delivery or request task
cr_w	Coverage of transit station s_w
L_{wt}	Grid where courier c_w is in period t

2.1 Preliminary

Partition the city into uniform grids to obtain an $I \times J$ grid map as Fig. 3 B). Each grid is hundreds of meters wide and long, thus not large for a delivery van.

Def. 1. Delivery task. A delivery task is a three-entry tuple $d_w = (d_w.s, d_w.g, d_w.\tau)$ describing that a parcel arrived at transit station $d_w.s$ was delivered to grid $d_w.g$ at time $d_w.\tau$.

Def. 2. Service task. A service task is a two-entry tuple $r_w = (r_w.g, r_w.\tau)$ meaning that a pick-up request was generated at timestamp $r_w.\tau$ with a service location in grid $r_w.g$; the waiting time of each request cannot be longer than a threshold ϑ.

Def. 3. Coverage. Coverage area of a transit station s_w is made up by grids to which the parcels arrived at s_w are delivered. It is denoted as cr_w and can be obtained by Eq. 1.

$$cr_w = \{d_k.g | d_k.s = s_w, k = 1, 2, \dots\} \quad (1)$$

Coverage frequency of s_w is a vector $cf_w \in R^{I \times J}$, where $I \times J$ is the number of grids in the city; each entry of cf_w is determined by Eq. 2 and 3, i.e. how many parcels arrived at s_w were delivered to each grid. By normalizing cf_w, we obtain the coverage distribution of station s_w and denote it as $cd_w \in R^{I \times J}$.

$$(cf_w)_v = |F_{wv}|, v = 1, 2, \dots, I \times J \quad (2)$$

$$F_{wv} = \{d_k | d_k.s = s_w, d_k.g = g_v, k = 1, 2, \dots\} \quad (3)$$

Problem Statement. Given historical delivery and service tasks per day, we try to learn an efficient courier management policy. It has the following two properties; firstly, instead of largely depending on hiring more couriers, it only adopts a portion of the currently hired ones; secondly, it maximizes the total number of completed tasks in each episode. Our policy has two steps, i.e. allocate parcels to each courier at transit station; guide where should each courier deliver and serve in each period. In our work, an episode is a specific time interval in the day, e.g. 8:00am – 1:00pm, 2:00pm – 7:00pm, or even the whole day; each period is set as 20 minutes.

2.2 Model Framework

Our model includes three components, i.e. city division into independent regions, parcel allocation and online courier management inner each of them.

City division based on connected component detection. Considering the large number of parcels and requests per day, as well as the hundreds or even thousands of couriers in a system, we firstly divide the city into independent regions, i.e. partition transit stations into groups G_1, G_2, ..., G_M such that their coverage areas do not overlap; here the coverage area of a group of stations G_w is as Eq. 4. Consequently, the city is divided into M independent regions C_1, C_2, ..., C_M each of which has their own set of transit stations and couriers. Afterwards, we respectively focus on each of them without considering the others outside.

$$C_w = \bigcup_{s_k \in G_w} cr_k \quad (4)$$

BDSB for parcel allocation at transit station. Before delivering parcels, couriers should firstly load them at transit station. In each specific region, BDSB determines which parcels are dispatched as a batch to a courier. In this step, multiple factors are considered, i.e. parcel locations, expected requests in the future, delivery-service burden, and the number of couriers in the region. As a soft-label clustering algorithm, BDSB gives labels to each grid, then parcels are dispatched based on labels of the grids where they locate, i.e. those located in grids of the same cluster are dispatched together to one courier; parcels in grids with soft labels are dispatched proportionally.

CCRL to manage couriers in real time. Online courier management is to guide where should each courier go and work in each period. Considering the random factors in practical process, i.e. stochastically coming requests and noise, generating a sequence of actions in advance for couriers to conduct one by one is not reasonable. Motivated by reinforcement learning theory, we propose a model to give actions to each courier in real time based on their real-time observations. To reduce the action space, we formulate our model in a multi-agent way, i.e. at each period, each courier gets their action one by one, considering those actions that some of their colleagues before them already got. Furthermore, we consider system context by action pruning, making our model more reasonable and efficient.

3. METHODOLOGY

Methodologies of each component are elaborated in this section. Real-world data from Beijing are adopted for examples.

3.1 Connected Component based City Division

City division is necessary to reduce the problem complexity. Besides, there is no need to consider the entire city at the same time, as couriers in two areas far away are impossible to interact.

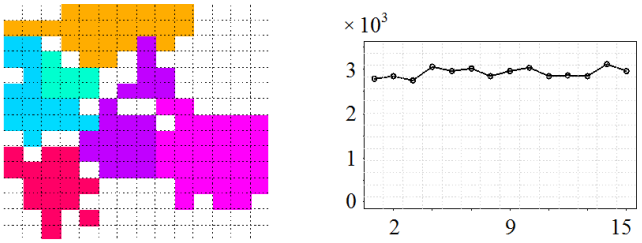
Def. 4. Independence. Stations s_w and s_v are independent if the distance of their coverage distributions is large, i.e. $|cd_w - cd_v| > 2 - \tau$, where $\tau \in (0, 2]$ is a parameter.

Station independence means that two stations rarely have overlap between their coverage areas, thus there is no interaction among their couriers. According to historical data, majority of the transit stations are not independent from each other, which is caused by the location of warehouses in the system and very hard to change. Ignoring the overlap and respectively focusing on each station is not efficient as couriers can cooperate when they work in overlapped areas. Motivated by this observation, we partition transit stations in the city into groups whose coverage areas rarely overlap; then a constant number of couriers can work cooperatively for each of them without considering the others. Station partition is very intuitive that based on Connected Component Detection [18] as follows.

- For each station s_w, we estimate its coverage distribution cd_w.
- If two stations s_w and s_v are not independent, we connect them by an edge. Consequently, a graph as Fig. 5 A) is obtained, where each node means a transit station.
- Detect the connected components [18] in the graph. Stations in the same component are partitioned into one group.

Obtaining station groups G_1, G_2, ..., G_M, we can divide the city into independent regions C_1, C_2, ..., C_M by Eq. 4. Some regions in Beijing are shown in Fig. 4 A), where each color denotes a region and the blank grids mean that they do not belong to these regions, i.e. they are in other regions, or there is no task in them because of some geographical reasons, e.g. it is a lake there. Afterwards, we can focus on each region respectively. If a single region is too small, we can combine multiple ones to obtain a hyper region and require couriers in them to cooperate, thus further improve the efficiency. Combining regions is reasonable according to Lemma

1, which is intuitive. In the following work, we focus on the hyper region made up by the six independent ones in Fig. 4 A).



A) Regions in Beijing B) # parcels per day

Fig. 4. Regions and task stability

Lemma 1. If $C_1 \perp C_2 \perp \cdots \perp C_M$, then $\bigcup_{k\in M_1} C_k \perp \bigcup_{k\in M_2} C_k$ when M_1, $M_2 \subset \{1, 2, \ldots, M\}$ and $M_1 \cap M_2 = \emptyset$; here $\perp$ means being independent with each other.

Each region has a set of stations. However, we consider their parcels together when allocating them to couriers. We can do this because each region is not too large, thus couriers can load parcels assigned to them from several stations in it quickly before departing to deliver. Compared with the length of an episode, e.g. half day or even one day, this time is very trivial.

3.2 BDSB for Parcel Allocation

In each region, how to dispatch parcels to couriers depends on multiple factors. Firstly, we need to consider the expected requests that may come later. Request expectation can be easily learned from historical data. As we observe that the number of requests is comparatively smooth in the day without any sudden peak nor collapse, we assume an equal request expectation in each period of the episode for simplicity. Secondly, it is better that the parcels dispatched to one courier locate in neighboring grids which is defined as Def. 5; this is to ensure efficiency in online management later. Lastly, we try to ensure that all couriers have almost even delivery-service burdens, thus to guarantee fair revenues, reduce labor waste and avoid overwork.

Def. 5. Neighboring grids. Given a set of grids A, if from any grid $g_w \in A$, a courier can go to another one $g_v \in A$, without visiting a third grid g_k that $g_k \notin A$ and it is not expected to have any request later, we define this set of grids as neighboring ones.

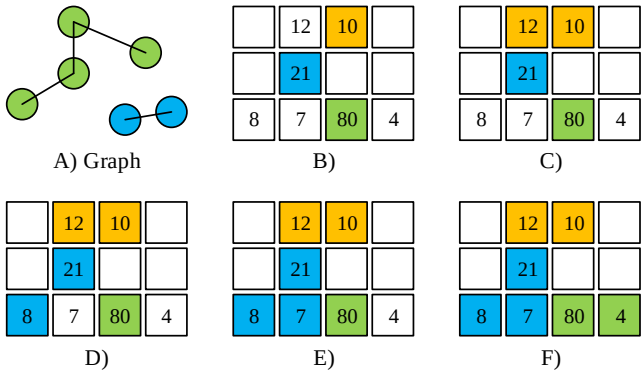
BDSB is proposed for parcel allocation. Main idea of BDSB is to softly cluster grids in the region into groups, then parcels in the grids of a same cluster are dispatched as a batch to one courier. Although BDSB can be conducted each time parcel allocation is required to obtain a specific result for the current episode, we adopt historical averages over episode to run it once, thus obtain a common clustering result shared by days for simplicity. We can do this for the following reasons. Firstly, according to historical data, the number and distribution of parcels and requests are stable from day to day. Fig. 4 B) shows the total number of parcels per day in the hyper region in Fig. 4 A), confirming our claim. Therefore, historical averages are representative enough. Besides, this step only aims to give a good initialization for the next step instead of a final optimal strategy, thus some turbulence beyond the averages can be ignored. BDSB has two iterative components, i.e. Primary Neighbor Clustering and Balance Improvement Clustering. Assuming there are n couriers in the region, i.e. the grids in it need to be clustered into n groups, we elaborate the two components of BDSB one by one.

Primary Neighbor Clustering (PNC). PNC simultaneously considers the neighboring property and delivery-service burden, giving a primary clustering result. It has the following six steps.

- Denoting the average number of parcels and requests at each grid g_i in the episode as p_i and q_i respectively, then we can estimate the work burden of each grid as $b_i = p_i + q_i$.
- Randomly select n grids in the region, each of which is considered as an initial cluster.
- Select the cluster A_k whose total work burden is minimum and the cluster A_v whose total burden is maximum.
- For A_k, denoting the grids surrounding it and not in any cluster yet as Λ_k, then select a grid $g_w \in \Lambda_k$ by Eq. 5.

$$\arg_{g_w\in\Lambda_k} \min |b_w + \sum_{g_j\in A_k} b_j - \sum_{g_j\in A_v} b_j| \quad (5)$$

- If g_w exists, update A_k to $A_k = A_k \cup g_w$; otherwise, replace A_k by the cluster with less minimum burden and go to step 4.
- Iterate step 3-5 until that there is no grid not in any cluster.



A) Graph B) C) D) E) F)

Fig. 5. Connected components and PNC algorithm

A running example is given in Fig. 5 to illustrate the above algorithm. Fig. 5 B) shows the work burden of each gird, where the ones without a number mean that they are not in this region. Assume that there are three couriers, thus the grids are to be clustered into three groups. Randomly select three grids to be the initial clusters as shown in Fig. 5 B), where each color means a cluster, i.e. A_1 – yellow, A_2 – blue, A_3 – green. Select the cluster with minimum and maximum total burden, i.e. A_1 and A_3. For A_1, its surrounding grid in the region which does not belong to any cluster yet and satisfies Eq. 5 is that with a burden of 12, therefore, we update A_1 to the one shown in Fig. 5 C). Repeat this step until Fig. 5 D); currently, the clusters with minimum and maximum burden are A_1 and A_3, however, as we can see, A_1 has no more surrounding grid in the region that is not in any cluster, thus we consider cluster A_2 which has a less minimum burden than A_1; the grid with a burden of 7 satisfies the constraints in step 4, therefore, we update A_2 to the one shown in Fig. 5 E). Now, as neither A_1 nor A_2 has any surrounding grid that is not in any cluster, they cannot be updated. Consequently, the grid with a burden of 4 is clustered to A_3 as in Fig. 5 F).

PNC gives a hard label to each grid, i.e. each grid belongs to only one cluster. However, as there are grids whose burdens are extremely large, hard labels cannot guarantee that all clusters have almost even work burdens, e.g. the burden of each cluster in Fig. 5 F) is 22, 36 and 84; therefore BIC is proposed to improve the hard-label clustering result to a soft-label one.

Balance Improvement Clustering (**BIC**). BIC keeps reallocating a portion of work burden from the cluster whose burden is too large to the one whose burden is small; to each grid, based on the percent its work burden is shared by each cluster, we generate soft labels for it. BIC has three steps.

- Select the cluster with maximum work burden and denote it as A_v; then select cluster A_k whose work burden is minimum among the surrounding clusters of A_v.
- Allocate a portion of work burden a_{vk} from cluster A_v to A_k while not violating their neighboring property; here a_{vk} is determined by Eq. 6.

$$a_{vk} \leq \frac{1}{2} \times |\sum_{g_j \in A_v} b_j - \sum_{g_j \in A_k} b_j| \quad (6)$$

- Iterate step $1-2$ for a given number of times. Generate soft labels for each grid by normalizing its burden in each cluster.

A running example is given in Fig. 6 to elaborate BIC. After obtaining Fig. 5 F), we select the cluster whose burden is maximum, i.e. A_3 with a burden of 84, and its surrounding cluster whose burden is the minimum, i.e. A_2 with a burden of 36. According to Eq. 6, we allocate a burden of $\frac{1}{2} \times (84-36) = 24$ from A_3 to A_2 as shown in Fig. 6 A). In the second iteration, we select A_2 as the cluster with maximum burden and A_1 from its surrounding ones whose burden is minimum; by Eq. 6, a burden of 19 is allocated from A_2 to A_1 as shown in Fig. 6 B). Assume that only two iterations are required, we generate labels for each grid as Fig. C). BDSB ensures a good result according to Lemma 2.

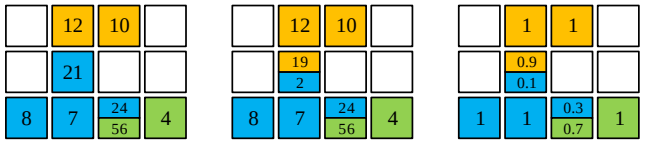


Fig. 6. BIC algorithm

Lemma 2. After enough iterations, BDSB can guarantee that all clusters have equal delivery-service burdens.

When allocating parcels, those located in grids of the same cluster are dispatched together to one courier; parcels in grids with soft labels are dispatched proportionally. Remind that even-burden is only a constraint when dispatching parcels, but not required any more in the following step; there are two reasons for this setting. Firstly, our main target is to complete more tasks. Secondly, we constrain the working area of each courier to their corresponding clusters, thus the unbalance will not be very severe, i.e. we can still ensure almost fair revenues, reduce labor waste and avoid overwork.

3.3 CCRL for Online Courier Management

After loading parcels at transit station, couriers choose where to go and work in each period. Considering the random factors in practical operation, and the target to maximize the total number of completed tasks in a long time, we propose a reinforcement learning based model to guide them in real time.

3.3.1 *Multi-Agent Reinforcement Learning*

A RL model has six components, i.e. $(\mathbb{S}, A, TR, R, \pi, \gamma)$, where $\mathbb{S}$ is the state set; A means the action space; TR describes the transition probability that an agent took action a_t under state S_t will transit to the next one S_{t+1}, i.e. $\mathbb{S} \times A \times \mathbb{S} \to TR$; R stands for the immediate reward received after taking an action under a state and transiting to the next one, i.e. $\mathbb{S} \times A \times \mathbb{S} \to R$; π is a policy $\mathbb{S} \times A \to \pi$, which describes the probability to take an action under a state; γ is a discount parameter. At each period t, an agent under state S_t takes an action a_t according to policy π, then transits to the next state S_{t+1}, receiving an immediate reward r_t. Each action has a long-term return as Eq. 7 where T is the last period in the episode.

$$U_t = r_t + \gamma \times r_{t+1} + \gamma^2 \times r_{t+2} + \cdots + \gamma^{T-t} r_T \quad (7)$$

Eq. 8 defines the optimal long-term value function, which describes the maximum expected long-term return of each action a_t under each state S_t, by following any policy after t. After obtaining this function, the corresponding optimal policy of the RL can be easily inferred by Eq. 9, i.e. always take the action with maximum optimal long-term value under the current state.

$$Q(S_t, a_t) = \max_\pi E_\pi[U_t | S_t, a_t, \pi] \quad (8)$$

$$a_t^* = \arg_a \max Q(S_t, a) \quad (9)$$

Bellman equation as Eq. 10 is usually adopted to estimate the optimal long-term value function via an iterative approach.

$$Q(S_t, a_t) = E_{S_{t+1}}[r_t + \gamma \times \max_a Q(S_{t+1}, a) | S_t, a_t] \quad (10)$$

As there are multiple couriers working in each region simultaneously, our model CCRL based on RL theory is formulated in a multi-agent way, i.e. couriers in the same region share a common model; therefore, at each period, these couriers get their actions one by one with a common policy based on their own specific states. Multi-agent setting can largely reduce the action space, i.e. from 9^n to 9, where n is the number of couriers in the region. However, while achieving high efficiency, how to make these agents interact cooperatively is a nontrivial challenge. Our model deals with the cooperation issue among couriers by designing the state in a novel way. Besides, region context is considered by pruning rules, further improving model efficiency.

3.3.2 *Contextual Cooperative Reinforcement Learning*

At each period t, we generate actions for couriers in the same region sequentially, i.e. instead of respectively focusing on each of them and generating an action without considering the others, we firstly consider c_1 and generate an action for her; based on this assigned action to c_1, we generate an action for c_2; then there comes the action for c_3 which is dependent on those of c_1 and c_2; repeat these steps until all the n couriers in this region have obtained their actions in t; then they conduct their actions simultaneously and transit to their next states in period $t+1$. Generating actions in a sequence instead of a parallel way aims at

better guaranteeing the cooperation among couriers. How to design each component in CCRL is as follows.

Agent. Couriers in one region are homogeneous agents; they choose where to go and work in each period by a common policy.

Action. An action $a_t \in R^9$ describes which grid the courier chooses to go at the beginning of period t.

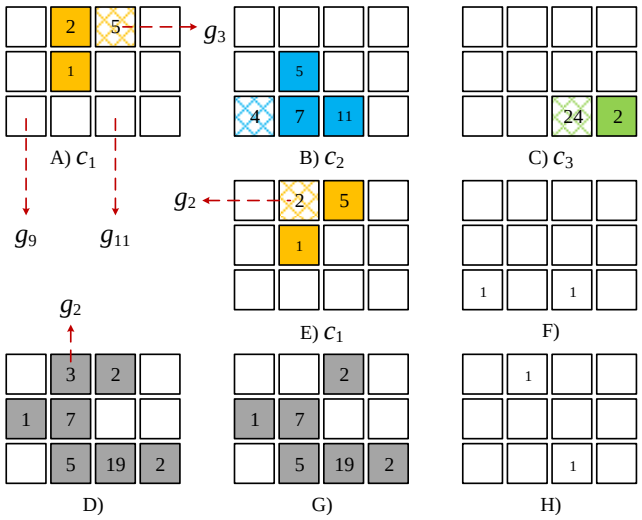


Fig 7. State designation in CCRL

State. At each period t, a courier c_w has a state consisting of two components as $S_{wt} = (S^1_{wt}, S^2_{wt})$, where S^1_{wt} means the global state while S^2_{wt} denotes the local one. Global state $S^1_{wt} = (\Upsilon_{wt}, W_{wt})$ describes the already came but unserved requests by Υ_{wt} and where the other couriers are by W_{wt}. For simplicity, we represent Υ_{wt} and W_{wt} by matrix corresponding to the grid map of the region, e.g. Fig. 4 A), where each entry corresponding to a grid in the region means how many unserved requests or couriers are there; to entries corresponding to grids not in this region, we set them as zero. Another benefit of matrix representation here is to make Convolutional Neural Network applicable later if necessary. Local state $S^2_{wt} = (D_{wt}, L_{wt}, t)$ describes the remaining parcels of the specific courier by D_{wt}, her current location by L_{wt}, and the current time by t; D_{wt} is represented by a matrix which is similar with Υ_{wt} and W_{wt} while L_{wt} and t are all represented by a one-hot vector. It is intuitive that local state S^2_{wt} varies over c_w in the same period t. Besides, as actions are generated in a sequence way and each courier is impacted by the already determined actions of other couriers before them, global state S^1_{wt} varies over c_w in the same period as well.

Fig. 7 gives an example to elaborate how to design the state of each courier in period t. Assuming there are three couriers whose remaining parcels are respectively shown in Fig. 7 A), B) and C); each shaded grid means the current location of the corresponding courier, e.g. c_1 is in grid g_3 now. Fig. 7 D) describes the currently unserved requests. For c_1, the unserved request matrix in Fig. 7 D) and the distribution of the other two couriers c_2 and c_3 in Fig. 7 F), i.e. one in g_9 while the other one in g_{11}, make up her global state; the remaining parcel matrix of c_1 in Fig. 7 A), her current location g_3, and the current time t, make up her local state. Based on this state, an action is assigned to c_1; we assume the assigned action is *go to the left grid*, i.e. g_2 as shown in Fig. 7 E). After c_1 got her action, we consider c_2. Firstly, we approximate how many requests are **expected** to be served by c_1 inner g_2 in t after she conducts the assigned action; assume this approximation as δ_1 = 3 in our example. Update the unserved request matrix in Fig. 7 D) by reducing δ_1 from the entry corresponding to g_2; we obtain the unserved request matrix for c_2 as in Fig. 7 G). Generate the distribution of other couriers for c_2 as Fig. 7 H); where the location of c_1 is updated to g_2 considering her action. Consequently, Fig. 7 G) and Fig. 7 H) make up the global state of c_2 while her local state is made up by her remaining parcel matrix in Fig. 7 B), her current location g_9, and the time t. Repeat these steps until all couriers are considered.

Immediate reward. After taking an action at the current state and transiting to the next one, a courier obtains a reward r_t, which means the total number of tasks she completed in period t.

After formally formulating CCRL, we design a Deep Neural Network to estimate its optimal long-term value function by Eq. 10. As discussed in section 3.2, the working area of each courier has been constrained to their corresponding clusters obtained from BDSB; which implicitly considers the geographical context of the region. Consequently, when some couriers are in some grids, they can only choose actions from a portion of all the nine possible ones. Instead of requiring the optimal long-term values corresponding to some state-action pairs to be zero, we prune the actions that will lead couriers to go out of their clusters directly. Pruning rules can largely improve the model training efficiency, which will be discussed in experiments. System simulator for model training and evaluation is elaborated in Appendix.

4. EVALUATION

Experiments on road network data and historical express data from Beijing are conducted to confirm the outperformance of our model. Express data are provided by one of the largest e-commerce platforms in China. Tab. 2 gives the data statistics.

Tab. 2. Real-world data

Time Duration	1, Aug. – 15, Aug. 2018
# range	≈ 15 × 15 km^2
# grids	30 × 30
# transit stations	106
# couriers	1,786
# parcels	531,920

4.1 Metric and Baselines

Metric. Percent of Completed Tasks (PCT) in an episode.

Baselines. Eleven baselines are adopted; the first two are clustering algorithms to confirm the outperformance of BDSB while the remaining ones are for online management. Baselines 3−6 are heuristic algorithms considering the *context* of the region and 7−11 are based on reinforcement learning theory.

PNC. Component 1 in BDSB algorithm.

PC. Do not consider the neighboring property in PNC.
Random. Select a random action for each courier at each time.
Value Greedy (VG). Conduct random policy for many times and calculate the value of each grid in each period by the averages of their immediate rewards. In each period t, a courier chooses the action whose value is the largest in t.
Value Softmax (VS). Similar with VG; but in each period each courier chooses the action based on the Softmax probability estimated from the average values instead of a greedy way.
Top$-k$. For each courier, estimate how many remaining tasks each grid has, and randomly choose one from the top$-k$ ones.
Q-Learning. Standard Q-Learning [5][17] where a value table is learned for online guidance; the state reduces to (L_{wt}, t).
Only parcel (OP). Only consider the remaining parcels of each courier; the state of each courier at period t is $(W_{wt}, D_{wt}, L_{wt}, t)$.
Only request (OR). Only consider the currently unserved requests; the state of each courier at period t is $(\Upsilon_{wt}, W_{wt}, L_{wt}, t)$.
Independent DQN (IDQN). Do not consider the cooperation among couriers; therefore, the state of each courier at period t is $(\Upsilon_t, D_{wt}, L_{wt}, t)$; here Υ_t is shared by all couriers in t.
No Context (NC). Do not consider the context of the region; therefore, no pruning rule are adopted when generating actions.

4.2 Evaluation Results

Experiment results for the hyper region in Fig. 4 A) are detailly analyzed in this section. We respectively set the number of couriers in it as $n = 40, 60, 80$, which correspond to no more than $35, 50, 65$ percent of the currently hired ones in it. We consider the morning in the day as an episode, i.e. 8:00am–1:00pm. Experiments in other regions and episodes are also conducted; as the results are similar, we do not discuss them here.

An Operation Trick. In each episode, we allocate little more parcels to couriers than those should be delivered, e.g. 8:00am–1:00pm corresponds to 0.4 day time, but we allocate 50 instead of 40 percent of all the parcels in the day to couriers for this episode. Because couriers may not complete all the loaded parcels in each episode, this setting avoids parcel accumulation, e.g. 50 percent of the parcels are dispatched, of which 80 percent are completed in the episode, thus $50 \times 80\% = 40$ percent of all the parcels in the day are delivered to their customers in 0.4 day. Based on this setting, we guarantee that our model can averagely complete all the delivery tasks per day. Otherwise, more and more parcels will accumulate at the transit station, which is not reasonable as parcels arrived at transit station must be delivered.

4.2.1 BDSB Clustering Results

Fig. 8 shows the clustering results of BDSB or PNC or PC. Fig. 8 A) − C) show the delivery-service burden of each cluster obtained by BDSB, respectively under each given number of couriers, i.e. $n = 40, 60, 80$. As we can see, almost even work burdens among clusters can be guaranteed. Relative standard deviations of the burden under these three conditions are respectively 0.029, 0.041, 0.046, which are very small. According to Lemma 2, all the clusters can have an equal delivery-service burden after enough iterations, however, after a given number of iterations, we pause the algorithm and return the result. We do this for two reasons; firstly, this step only aims to give a good initialization, thus it is not necessary to ensure an exact balance; besides, we pause earlier thus the coverage area of each cluster is not large, which is preferred in practical operation.

Fig. 8 D) shows the delivery-service burden of each cluster obtained by PNC when $n = 40$. As we can see, PNC cannot ensure the work burden balance property satisfactorily, thus the second component BIC in BDSB is necessary. Fig. 8 E) – F) show the burden of each cluster obtained by PC when $n = 40$ and 60 respectively; their unbalance issue is less severe than those of PNC, which is intuitive because PNC also tries to ensure the neighboring property. However, PC still cannot perform well enough, especially when n is large, i.e. $n = 60$ in Fig. 8 F), leading to unfair revenues, labor waste and overwork.

Fig. 8 G), H) and L) show some clusters obtained by BDSB when $n = 40$, confirming that BDSB can guarantee the neighboring property.

4.2.2 CCRL Online Management Results

Obtaining clusters by BDSB / PNC / PC, we conduct online courier management by nine baselines and our model; experiment results are summarized in Tab. 3, where the PCT of delivery, request and ALL tasks are summarized.

As we can see, more delivery tasks can be completed than the request ones when n is small while more requests can be served when n becomes large; this is reasonable as we always ask couriers to deliver parcels first in each period; when the number of couriers increases, more labor can be devoted to serve the pick-up requests. Because delivery tasks in one grid can only be conducted by the couriers who have loaded its parcels while the request tasks can be served by any one working inner it, it is reasonable that more requests can be completed when n is large.

According to Tab. 3, the first four heuristic algorithms cannot work satisfactorily; this is intuitive as they consider neither cooperation among couriers nor the long-term optimization target. VG can rarely improve Random or even performs worse when $n = 60$ and 80 while VS has obvious improvement; this is because the system is very dynamic, thus historical values cannot be totally trusted. However, they can be partially referred to, therefore, VS which combines historical values with random policy performs the best among these three algorithms. Top$-k$ performs the best among all heuristic algorithms, as it has a myopic optimization target, besides, it depends on real-time observations, which is more suitable to a dynamic system.

Baselines $5 - 9$ confirm the necessity of each component in CCRL model. As a traditional RL method, Q-learning cannot address the large space issue, thus can only consider limited information of the environment, leading to a poor performance. OR does not consider parcel information, thus can complete a little bit more service tasks; this is possible as the model may make couriers sacrifice some delivery tasks to complete more service ones. However, as our target is to complete more total tasks, OR is not good enough, neither is OP which does not consider the pick-up requests. IDQN focuses on each courier in a parallel way and does not consider the cooperation among them; therefore, its performance is worse than our model. NC also

performs poorly as it does not incorporate prior knowledge in model training, leading to slow convergence, or even bad results.

Tab. 3. PCT of each model with a given number of couriers

# Couriers	40			60			80		
	Parcel	Request	All task	Parcel	Request	All task	Parcel	Request	All task
Random	0.619	0.482	0.552	0.717	0.694	0.705	0.752	0.829	0.790
Value Greedy	0.553	0.653	0.602	0.555	0.692	0.622	0.567	0.820	0.691
Value Softmax	0.687	0.657	0.672	0.712	0.791	0.751	0.726	0.886	0.804
Top–k	0.766	0.596	0.684	0.808	0.757	0.783	0.802	0.880	0.840
Q-Learning	0.719	0.700	0.710	0.731	0.822	0.775	0.785	0.916	0.849
Only Parcel	0.802	0.695	0.750	0.831	0.836	0.833	0.811	0.919	0.864
Only Request	0.776	**0.740**	0.758	0.760	**0.859**	0.809	0.768	**0.935**	0.850
Independent DQN	0.794	0.666	0.731	0.801	0.821	0.811	0.814	0.903	0.857
No Context	0.733	0.634	0.684	0.758	0.747	0.752	0.723	0.790	0.756
BDSB + CCRL	**0.812**	**0.715**	**0.765**	**0.835**	**0.846**	**0.841**	**0.825**	**0.925**	**0.874**

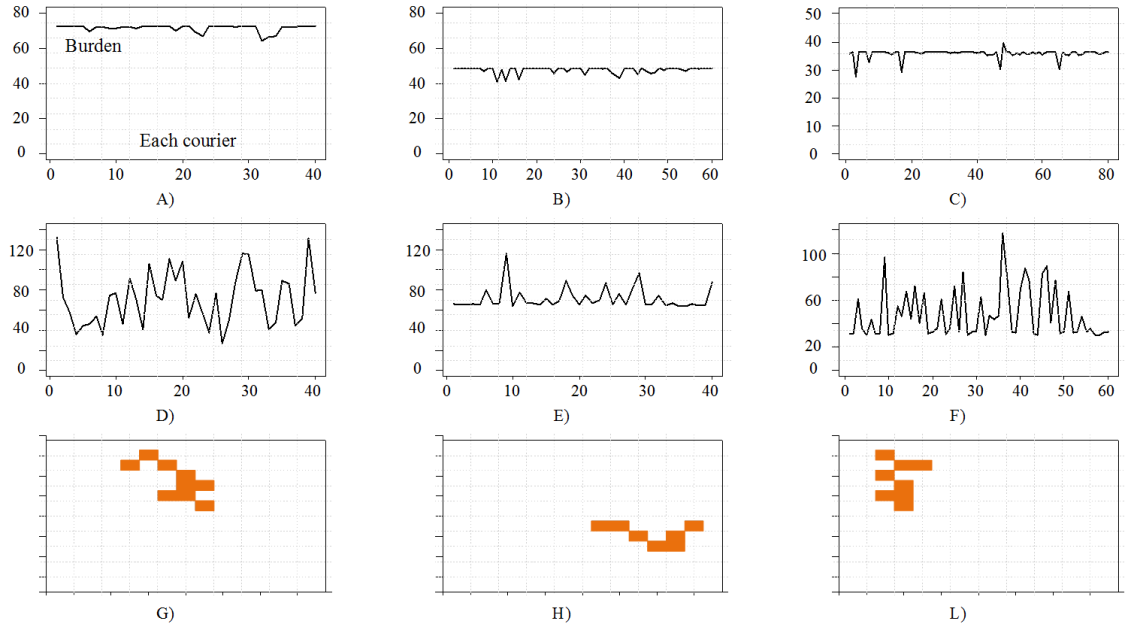


Fig. 8. Clustering results with BDSB / PNC / PC algorithm

4.2.3 *Case Study*

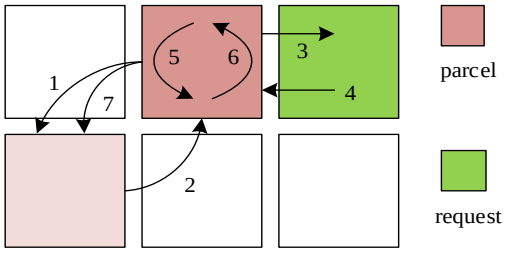


Fig. 9. Sequence of actions of a courier

Fig. 9 shows a sequence of actions of a specific courier, where an arrow with a number means the action order, red grids show the locations of her remaining parcels while the green ones describe hot request grids around her; the darker the color is, the more tasks there are. As we can see, when choosing the next action, there is no priority between delivery and service, i.e. the courier does not try to complete any of them firstly but conduct them alternatively; this is reasonable as our target is to complete more tasks instead of any one of them. Besides, we can see that the courier does not work greedily, i.e. always chooses the grid with the most tasks; we think she works in this way to achieve a long-term instead of myopic good result.

Cooperation among couriers are also analyzed, i.e. will couriers go to the same grid and compete for tasks. According to experiments, we observe that couriers often go to the same grid under three scenarios; the first one is as Fig. 10 where c_1 and c_2 both go to g_1 in the same period; Fig. 10 A) and B) respectively show the remaining parcels of them; Fig. 10 C) show the unserved requests around them. Although both c_1 and c_2 go the g_1, there is no competition between them, because c_1 goes to g_1 for requests

while c_2 goes there to deliver as she has remining parcels in g_1. Another scenario is that multiple couriers go to a same grid but neither of them has remaining parcels there. In this condition, there are often many requests in that grid waiting for service, thus these couriers work together there instead of competing. Last scenario is that several couriers have their own parcels in a same grid, thus they work separately there without competing.

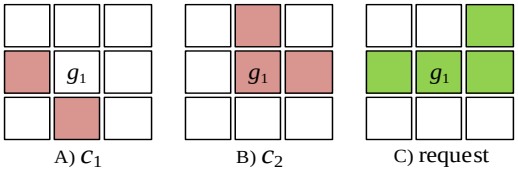


Fig. 10. Cooperation among couriers

5. RELATED WORK

Express System and Operation. Express systems are widely deployed in many major cities and generating massive express data. Zhang et al. [4] systematically studied the large-scale dynamic city express problem and proposed some heuristic algorithms. However, no real-world express data are analyzed nor used in their work. Besides, they gave not only each request, but also each parcel a deadline, which is not often the case in practical operation; random noise was not considered neither. Some studies tried to adopt crowdsourcing to deliver parcels to customers, e.g. Sadilek et al. [3] asked a group of twitter users to deliver parcels; Mclnerney et al. [2] employed mobile users for delivery; Chen et al. [1] exploited existing taxi service to deliver packages to their destinations. As our problem is to effectively manage the hired couriers for delivery and pick-up service, these previous works cannot be adopted to our problem directly. Besides works on express system, many studies of operation on spatio-temporal systems are conducted as well. Lin et al. [5] gave a solution to the large-scale fleet management problem for ride-sharing platforms. Wei et al. [8] focused on how to intelligently control the traffic light such that the average waiting time of each vehicle is minimum. For a bike-sharing system, there are multiple studies [7][9][12][14] about how to reposition bikes among stations in a city to reduce the customer loss. As a specific operation problem with specific formulation, we cannot adopt these previous models in our work directly.

Deep Reinforcement Learning. Because of the large state and action spaces in many practical problems, DRL is proposed to utilize deep neural networks for function approximation in the traditional RL model, which significantly improve the performance of many challenging applications [10][11][13]. Currently, some studies [5][7][8] try to adopt DRL to solve practical problems on spatio-transit data. Besides, models applying DRL to recommendation is proposed as well; e.g. Chen et al. [15] proposed a robust DQN method to gain better recommendation performance in a dynamic e-commerce platform; Hu et al. [16] adopted DRL to learn an optimal ranking policy for each search, etc. However, these models cannot be directly adopted to our problem neither.

6. CONCLUSION

In this paper, we propose a reinforcement learning based framework to manage the couriers in an express system. We firstly divide the entire city into independent regions, each of which has a constant number of couriers. A BDSB clustering algorithm is then proposed to dispatch parcels to couriers at transit station in each region. Afterwards, couriers in each region are guided to deliver and serve by a common policy, which is learned by CCRL model. We confirm the outperformance of our model by experiments on real-world data from Beijing.

REFERENCES

[1] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, E. Sha. CROWDDELIVER: Planning City-Wide Package Delivery Paths Leveraging the Crowd of Taxi. IEEE Trans. Intelligent Transportation Systems, 2017.

[2] J. Mclnerney, A. Rogers, N. R. Jennings. Crowdsourcing Physical Package Delivery Using the Existing Routine Mobility of a Local Population. In Proc. Orange D4D Challenge, 2014.

[3] A. Sadlilek, J. Krumm, E. Horvitz. Crowdphysics: Planned and Opportunistic Crowdsourcing for Physical Tasks. In Proc. ICWSM, 2013.

[4] S. Zhang, L. Qin, Y. Zheng, H. Cheng. Effective and Efficient: Large-scale Dynamic City Express. Transaction on Knowledge and Data Engineering.

[5] K. Lin, R. Zhao, Z. Xu, J. Zhou. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In Proc. SIGKDD, 2018.

[6] N. Garg, S. Ranu. Rout Recommendations for Idle Taxi Drivers: Find Me the Shortest Route to a Customer. In Proc. SIGKDD, 2018.

[7] Y. Li, Y. Zheng, Q. Yang. Dynamic Bike Reposition: A Spatio-Transit Reinforcement Learning Approach. In Proc. SIGKDD, 2018.

[8] H. Wei, G. Zheng, H. Yao, Z. Li. IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control. In Proc. SIGKDD, 2018.

[9] P. Hulot, D. Aloise, S. D. Jena. Towards Station-Level Demand Prediction for Effective Rebalancing in Bike-Sharing Systems. In Proc. SIGKDD, 2018.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. Playing Atari with Deep Reinforcement Learning. arXiv, 2013.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis. Human-Level Control through Deep Reinforcement Learning. Nature, 2015.

[12] J. Liu, L. Sun, W. Chen, H. Xiong. Rebalancing Bike Sharing Systems: A Multi-source Data Smart Optimization. In Proc. SIGKDD, 2016.

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. v. d. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature, 2016.

[14] S. Ghosh, M. Trick, P. Varakantham. Robust Reposition to Counter Unpredictable Demand in Bike Sharing Systems. In Proc. IJCAI, 2016.

[15] S. Chen, Y. Yu, Q. Da, J. Tan, H. Huang, H. Tang. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In Proc. SIGKDD, 2018.

[16] Y. Hu, Q. Da, A. Zeng, Y. Yu, Y. Xu. Reinforcement Learning to Rank in E-Commerce Search Engine: Formalization, Analysis, and Application. In Proc. SIGKDD, 2018.

[17] R. S. Sutton, A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.

[18] J. Hopcroft, R. Tarjan. Algorithm 447: Efficient Algorithms for Graph Manipulation. Communication of the ACM, 1973.

Appendix - SYSTEM SIMULATOR

In this section, we design a simulator based on road network data and historical express data, to simulate how the system operates in each episode. Firstly, we estimate the driving distance between any two tasks in each grid based on the road network, e.g. assuming that any two road nodes e_v and e_k has a distance φ_{vk}, we estimate the task distance inner g_w with a normal distribution N_w by fitting all φ_{vk} where $e_v, e_k \in rn_w$ and rn_w is the road network in g_w. As parcels and requests are robust from day to day, we then adopt normal distributions learned from historical data to generate them at each grid.

An episode. Initially, our simulator generates parcels located in each grid for the entire episode; then BDSB allocates them to each courier. Afterwards, for each period t in this episode, we firstly update the unserved requests, i.e. those which are not served in a waiting time ϑ are deleted and those that may come in t are generated; then our simulator simulates the activities of each courier in t one by one. Each courier works as follows.

- Courier c_w conducts her action, i.e. goes to L_{wt} to work in t.
- If c_w has undelivered parcels in L_{wt} on her van and there is remaining time before the next period, generate a distance for c_w to go and deliver. Duration for this parcel delivery is estimated by Eq. 11, where $d_x \sim N_{wt}$ is the delivery distance; N_{wt} is the task distance distribution in grid L_{wt}; v_r is the speed of a delivery van; t_ε is a constant time needed by each task, e.g. the time for checking or form filling, etc.

$$t_x = \frac{d_x}{v_r} + t_\varepsilon \tag{11}$$

  Check whether the remaining time in t is enough for t_x, if yes, conduct this parcel delivery, update the remaining time in t by reducing t_x, update the remining parcels of c_w in L_{wt} by reducing one, and repeat step 2; otherwise, go to step 3.
- If there are unserved requests in L_{wt} and there are remaining time in t, generate a distance for c_w to go and serve. Duration for this service is also estimated by Eq. 11. Check whether the remaining time is enough for this service, if yes, conduct it, update the remining time, update the remining unserved requests in L_{wt} by reducing one, and repeat step 3; otherwise, c_w terminates her work in t.

Simulating the activities of each courier one by one matches the action generation process well, forcing each courier to consider the already determined actions of others enough.