

A COMPUTER-IMPLEMENTED METHOD, AND A SERVER

Technical Field

The invention relates generally to the field of logistics and/or transport. One aspect of the invention relates to a computer-implemented method. Another aspect of the invention relates to a server.

Background

In terms of logistics, it may be useful to optimise available delivery vehicles amongst a number of concurrent delivery bookings. Often the optimum allocation involves allocating a number of pickup location and drop off points for a single delivery vehicle to ensure the most efficient outcome overall.

In terms of transport, it may be useful to optimise available transport vehicle amongst a number of concurrent passengers. Often the optimum allocation involves including a number of nearby transport vehicles and passengers together in a batch and then solving the optimum allocation to minimise the waiting time for transport vehicles to reach passengers.

In order to achieve the optimum allocation, it may be necessary to divide a geographic territory into smaller areas in order to reduce the computational complexity for real time allocation and to reduce any latency in allocating vehicles. There is an inherent compromise in reducing the size of the pool of candidates to form batches to reduce latency and/or computational complexity, compared to the overall efficiency of the optimisation.

It may be desirable to provide a more effective or efficient solution in some applications.

Summary

Embodiments may be implemented as set out in any of the independent claims. Some optional features are defined in the dependent claims.

Implementation of the techniques disclosed herein may provide significant technical advantages. Advantages of one or more aspects may include one or more of the following:

- By expanding the candidate pool through bidirectional candidate retrieval, the algorithm's batching probability is increased, leading to better efficiency choices based on a wider range of batching combinations.
- The technical solution of faster processing of orders based on the technical problem of inefficient batching of candidates.
- The technical solution of reduction in the amount of communication traffic due to less cancelled orders based on the technical problem of inefficient batching of candidates.
- The technical solution of reduced data centre energy requirements based on the technical problem of inefficient batching of candidates.
- The technical solution of reduced greenhouse emissions based on the technical problem of inefficient batching of candidates. Because the orders are limited to those with the most efficient allocation, there may be less wasted trips by drivers, or less wasted sales by merchants, and therefore greenhouse emissions for any unnecessary trips or unnecessary product manufacturing may be avoided.
- The technical solution of reduced data centre storage requirements based on the technical problem of inefficient batching of candidates.
- The technical solution of reduced estimated time of arrival (ETA) for a given territory based on the technical problem of inefficient batching of candidates.
- The technical solution of improved transport experience by allowing drivers to arrive at customers at the minimum amount of time, avoiding waiting time for the customer and/or long travel times for the driver, based on the technical problem of inefficient batching of candidates.
- The technical solution of improved data quality and accuracy, based on the technical problem of inefficient batching of candidates.

- The technical solution of reduced server hardware required for the technical problem of inefficient batching of candidates. Reduced greenhouse emissions may result from less manufacturing of hardware.
- The technical solution of reduced bandwidth requirements based on the technical problem of inefficient batching of candidates. Reduced greenhouse emissions may result from less communications bandwidth requirements.
- The technical solution of less latency for the technical problem of inefficient batching of candidates.

In an exemplary implementation, the functionality of the techniques disclosed herein may be implemented in software running on a handheld communications device, such as a mobile phone. The software which implements the functionality of the techniques disclosed herein may be contained in an “app” – a computer program, or computer program product – which the user has downloaded from an online store. When running on the, for example, user’s mobile telephone, the hardware features of the mobile telephone may be used to implement the functionality described below, such as using the mobile telephone’s transceiver components to establish the secure communications channel.

In an exemplary implementation, the functionality of the techniques disclosed herein may be implemented in software running on a server communication apparatus (such as a one or more servers, one or more virtual machines, one or more processors or a cloud computing platform), which communicates with the applications running on the user terminals, driver terminals and/or merchant terminals such as mobile phones. The software which implements the functionality of the techniques disclosed herein may be contained in a computer program, or computer program product. The server communication apparatus establishes secure communication channels with the driver,

merchant and/or user terminals, for allocating users and/or merchants to drivers.

Brief Description of the Drawings

The invention will now be described, by way of example only, and with reference to the accompanying drawings in which:

Figure 1 is a schematic block diagram illustrating an exemplary delivery/transportation service.

Figure 2 is a schematic block diagram illustrating an exemplary communications server for the delivery/transportation service.

Figure 3 is a simplified flow diagram of proposed unified batching system

Figure 4 is a flow diagram of an existing sequential batching system Figure

5 is a flow diagram of proposed unified batching system

Figure 6 is an exemplary timeline of delivery bookings in an existing sequential batching system

Figure 7 is an exemplary timeline of delivery bookings in proposed unified batching system

Figure 8 is an exemplary timeline of delivery bookings in proposed unified batching system

Detailed Description

The techniques described herein are described primarily with reference to use in allocating delivery bookings to delivery vehicles. This may be useful to optimise overall efficiency.

Figure 1 shows an exemplary architecture of a system 100, with a number of users each having a communications device 104, a number of merchants each having a communications device 109, a number of drivers each having a user interface communications device 106, a server 102 (or geographically distributed servers) of a delivery platform provider and communication network 108 connecting each of the components. Each user contacts the server 102 using a user software application (App) on the communications

device 104. Similarly, the drivers and merchants may use an app on their devices 106, 109.

A platform provider may use the system 100 for the purpose of logistics and/or transport management. This may involve deliveries or e-commerce-based transactions, where a customer orders some food, a product, or other physical item needing delivery. This may involve transportation, where multiple adjacent customers may order a vehicle, where one vehicle is then shared to transport each customer to multiple adjacent destinations, or where each customer has their own vehicle. Typically, the system 100 used by the platform operator will facilitate transactions between the users, and goods or services providers such as drivers or merchants. The system 100 may then be used to simultaneously optimise the allocation of available delivery vehicles (including different applicable delivery vehicle types) for transport of delivery bookings and/or passengers.

For deliveries or e-commerce-based transactions, the user device 104 may allow the users to input queries containing the keywords for the items of interest and delivery addresses. The user may see a list of merchants and/or items provided by the merchants, and order items from the merchants. The merchant may contact the server 102 using the merchant device 109 for providing information about their items and receiving orders for each confirmed transaction. The driver contacts the server 102 using the driver device 106. The driver device 106 allows the drivers to indicate their availability to take the delivery jobs, information about their vehicle, their location. The server 102 may then match drivers to the delivery, based on, for example: geographic location of merchants and drivers, maximising revenue, user or driver feedback ratings, weather, driving conditions, traffic level / accidents, relative demand, environmental impact, and/or supply levels. The user may be offered a particular delivery cost and approximate delivery ETA. If the user accepts the offer, the server 102 may go through a payment authorisation process. If the authorisation is approved, the merchant will then

be notified and directed to provide goods for the driver to pick up. The selected driver will then be notified and directed to the pickup location to pick up the goods. The driver may be optimised to pick up goods from several merchants and drop off the series of orders in a single efficient path. During the delivery, the user device 104, the driver's device 106, the merchant's device 109 and the server 102 may be updated with real-time trip information including the real-time location of the driver's vehicle, the destination, the driver fare and/or other trip-related information. After the trip the driver's device 106 may send a confirmation that the trip has ended to server 102. Once the transaction is approved and/or the delivery is completed, the user device 104, the driver's device 106, the merchant's device 109 and the server 102 may be updated with details of the completed financial transaction. This allows an efficient allocation of resources because the available fleet of drivers is optimised for the users' demand in each territory.

For transportation, the user device 104 may allow the user to enter their pickup location, a destination address, one or more service parameters, and/or after-ride information such as a rating. The one or more service parameters may include the number of seats of the vehicle, the style of vehicle, level of environmental impact and/or what kind of transport service is desired. Each driver contacts the server 102 using a driver app on the communication device 106. The driver app allows the driver to indicate their availability to accept jobs, information about their vehicle, their location, and/or after-ride info such as a rating. The server 102 may then match users to drivers, based on, for example: geographic location of users and drivers, maximising revenue, user or driver feedback ratings, weather, driving conditions, traffic level / accidents, relative demand, environmental impact, and/or supply levels. The user may be offered a particular transport cost, or a range based on different types of vehicles, and an approximate ETA. If the user accepts the offer, the system may go through a payment authorisation process. If the authorisation is approved, the selected driver will then be notified and directed to the pickup location to pick up the user/passenger. This

may be optimised for several users wishing to share a vehicle to lower the trip cost, or single vehicles. During the trip, the user device 104, the driver's device 106, and/or the server 102 may be updated with real-time trip information including the real-time location of the driver's vehicle, the destination, the trip fare and/or other trip-related information. At the conclusion of the trip, the driver's device 106 may send a confirmation the trip has ended to server 102. Once the transaction is approved and/or trip completed the user device 104, the driver's device 106, and/or the server 102 may be updated with details of the completed financial transaction. This allows an efficient allocation of resources because the available fleet of drivers is optimised for the users' demand in each territory.

Referring to Figure 2, further details of the components in the system of Figure 2 are now described. The communication apparatus 100 comprises the communication server 102, and it may include the user communication device 104, the merchant communication device 109 and the driver communication device 106. These devices are connected in the communication network 108 (for example, the Internet) through respective communication links 110, 111, 112, and 114 implementing, for example, internet communication protocols. The communication devices 104, 106 and 109 may be able to communicate through communication networks and/or protocols, including cellular communication networks, LAN, WAN, private data networks, VPN, fibre optic connections, laser communication, microwave communication, satellite communication, Bluetooth, Wifi, NFC, etc., but these are not specified in Figure 2 for the sake of clarity.

In the example shown in Figure 2, the communication server apparatus 102 may comprise several individual components including, but not limited to, one or more microprocessors 116, one or more memories 118 (e.g., a volatile memory such as a RAM), and/or longer-term, non-volatile, or persistent, memory 119 (e.g., SSD (Solid State Drive) or Hard disk drives (HDD)) for the loading of executable instructions 120, the executable instructions defining the

functionality the server apparatus 102 carries out under the control of the microprocessor 116. The communication server apparatus 102 also comprises one or more input/output modules 122 allowing the server to communicate over the communication network 108. One or more user interfaces 124 are provided for administrator control and may comprise, for example, computing peripheral devices such as display monitors, computer keyboards and the like.

The communication server apparatus 102 may be a single server as illustrated schematically in Figure 2. Alternatively, the functionality performed by the server apparatus 102 may be distributed across multiple physically or logically separate server components. In the context of the present specification, "a server", "the server" or "said server" is a computer program that is running on appropriate hardware and is capable of receiving requests (e.g., from electronic devices) over a network, and carrying out those requests, or causing those requests to be carried out. The hardware may be implemented as one or more physical computers, one physical computer system, one or more virtual machines, or a cloud-based server network. In the present context, the use of the expression a "server" is not intended to mean that every task (e.g. received instructions or requests) or any particular task will have been received, carried out, or caused to be carried out, by the same server (i.e. the same software and/or hardware); it is intended to mean that any number of software elements or hardware devices may be involved in receiving/sending, carrying out or causing to be carried out any task or request, or the consequences of any task or request; and all of this software and hardware may be one server or multiple servers. The same approach is to be taken for references to other systems, software or hardware components e.g., processor, memory, storage etc.

Here the transactions referred to may be payments done for anything the platform provider offers, for e.g., Ride-hailing, Ride-sharing, food delivery, e-commerce deliveries, etc. Transactions can include any interaction where a

user, driver or merchant has to pay to the platform provider for its services or products, request transportation, accept orders or deliveries, or otherwise communicate with the platform provider.

The server apparatus 102 may also comprise one or more databases 126 stored in volatile memory 118 and/or non-volatile memory 119, for storing data, which may include data on merchant behaviour, geographic information, images, products, points of interest, users, drivers, merchants, transaction data, aggregated data or parameters, payment data, and other relevant data. The data may be stored in a data structure according to the requirements of the application, or as described in more detail below. The database 126 may be replicated, distributed, sharded or otherwise optimised according to the requirements of the application.

The user communication device 104 may comprise several individual components including, but not limited to, one or more microprocessors 128, a memory 130 (e.g., a volatile memory such as RAM), and/or longer-term memory such as flash memory or SSD (Solid State drives) for the loading of executable instructions 132, the executable instructions defining the functionality the user communication device 104 carries out under the control of the microprocessor 128. The user communication device 104 also comprises an input/output module 134 allowing the user communication device 104 to communicate over the communication network 108. A user interface 136 is provided for user control. If the user communication device 104 is, say, a smartphone or tablet device, the user interface 136 will have a touch panel display as is prevalent in many smartphones and other handheld devices. Alternatively, if the user communication device 104 is, say, a desktop or laptop computer, the user interface 136 may have, for example, computing peripheral devices such as display monitors, computer keyboards and the like.

The merchant communication device 109 may be, for example, a smartphone or tablet device with the same or a similar hardware architecture to that of the user communication device 104.

The driver communication device 106 may be, for example, a smartphone or tablet device with the same or a similar hardware architecture to that of the user communication device 104. Alternatively, the functionality may be integrated into a bespoke device such as a taxi fleet management terminal or other logistics terminals.

The driver mentioned above may also be implemented by a remote driver or an autonomous driver. Examples include autonomously driven vehicle, UAVs, and drones. In that case the merchant may be responsible for loading the order with the delivery vehicle, or that task may be automated as well.

Thus, it will be appreciated that Figures 1 and 2 and the foregoing description illustrate and describe a system 100 comprising:

- a communication server 102;
- at least one user communication device 104 having an associated user
- and configured to initiate an unallocated delivery booking which includes a pick up and drop off location;
- at least one driver communication device 106 having an associated
- delivery vehicle and configured to provide delivery vehicle location data; and
- communication network equipment 108 configured to establish communication with the communications server 102, at least one user communication device 104, and at least one driver communication device 106;
- wherein the communications server 102 comprises at least one

processor(s), at least one memory 118, the server being configured, under control of one or more of the at least one processor(s) 116, to execute instructions stored in one or more of the at least one memory 118 to:

- receive a plurality of delivery bookings;
- store data for the received delivery bookings in a candidate pool;
- determine an allocation of a plurality delivery vehicles to one or more of the delivery bookings in the candidate pool;
- store data relating to the allocated delivery bookings and unallocated delivery bookings in the candidate pool;
- redetermine an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool; and
- dispatch delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.

Further, it will be appreciated that Figures 3 and 5 illustrate and describe a method performed in a communication server apparatus 102, the method comprising, under control of a microprocessor 116 of the communication server apparatus 102:

- receiving 302 a plurality of delivery bookings;
- storing data for the received delivery bookings in a candidate pool 307;
- determining 304 an allocation of a plurality delivery vehicles to one or more of the delivery bookings in the candidate pool 307;
- storing data relating to the allocated delivery bookings 306 and unallocated delivery bookings 308, 309 in the candidate pool 307;
- redetermining 310 an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool 307; and

dispatching 312 delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.

According to a first example embodiment shown in Figure 4, a method 400 of allocating bookings to drivers is shown. The method 400 includes Upfront Batching 402 and In-transit Batching 426, and follows a sequential approach.

The sequential batching system 400 attempts to form an upfront batch via an Upfront Batching Engine 408 with delivery bookings from an upfront batching Candidate Pool 406 within a specified upfront pooling time window. Delivery bookings from the Upfront Batching Candidate Pool 406 may comprise new unallocated delivery bookings 404 and unbatched delivery bookings that have not exceeded the specified upfront pooling time 422.

If the upfront batching is successful 410, the batch will be dispatched 416 for Pick up 418 and Drop off 420.

Dispatched delivery bookings indicate that the delivery bookings, whether batched or unbatched, have been allocated to a delivery vehicle for Pick up and Drop Off.

If the upfront batching is unsuccessful 412, the system will perform a check 414 on the upfront pooling time against the upfront pooling time window. An upfront pooling time window is a specified time window which determines the maximum amount of time an unbatched delivery booking can stay in the upfront batching Candidate Pool 406 before being sent to In-transit Batching 426. If the upfront pooling time does not exceed the specified upfront pooling time window 422, the unbatched delivery booking will return to the upfront batching Candidate Pool 406 for another attempt to form an upfront batch. If the upfront pooling time exceeds the specified upfront pooling time window 424, the unbatched delivery booking will proceed to In-transit Batching 426 via an In-transit Batching Engine 430.

In In-transit Batching 426, the system will attempt to batch the unbatched delivery bookings 424 that failed Upfront Batching 402 with dispatched delivery bookings 416 that have not been picked up by their allocated delivery vehicles. These dispatched delivery bookings 416 could be dispatched batches 410, 432 from Upfront 402 and In-transit Batching 426, or dispatched solo delivery bookings that have exceeded their in-transit pooling time 438. For dispatched batches 410, 432, these batches will go through In-transit Batching 426 in their respective batched state, i.e., as a unified entity. A successful in-transit batching combines an unbatched delivery booking with a dispatched batch that has not been picked up, without reconfiguring any of the pre-existing delivery bookings within the dispatched batch.

If the in-transit batching is successful 432, the batch will be dispatched for Pick up 418 and Drop off 420.

If the in-transit batching is unsuccessful 434, the system will perform a check 436 on the in-transit pooling time against the in-transit pooling time window. An in-transit pooling time window is a specified time window which determines how long the unbatched delivery bookings can stay in in-transit batching before the system dispatches them as solo delivery bookings. If the in-transit pooling time does not exceed the specified in-transit pooling time window 440, the unbatched delivery booking will return to the In-transit Batching Engine 430 for another attempt to form an in-transit batch. If the in-transit pooling time exceeds the specified in-transit pooling time window 438, the unbatched delivery booking will be dispatched 416 for Pick up 418 and Drop off 420.

One of the limitations of the existing sequential batching system include lost opportunities for more efficient batches to be formed. The candidates for In-transit Batching 426 are restricted to the unbatched delivery bookings 424 from the Upfront batching 402 that have exceeded the specified upfront batching time window and dispatched batches 410, 432 from Upfront 402 and

In-transit Batching 426 that have not been picked up by their allocated delivery vehicles. As discussed above, these dispatched batches undergo in-transit batching in a batched state. Numerous potentially more efficient batches could be formed if these dispatched batches undergo batching with other unbatched delivery bookings in their respective unbatched state.

A second embodiment proposes a method 500, shown in Figure 5, with an innovative batching candidate retrieval approach that combines upfront batching and in-transit batching. This bidirectional candidate retrieval method 500 takes into account both previously dispatched (allocated to a delivery vehicle) delivery bookings and imminent incoming unallocated delivery bookings at the same time, enabling the formation of more efficient batches and allocation of delivery vehicles.

In terms of implementation, this method can be seamlessly integrated into the existing batching candidate retrieval and dispatch services with minimal alterations.

Apart from Food deliveries and Ride hailing business, this bidirectional candidate retrieval method can also be applied to the general logistic industry, where order batching is utilized to streamline the delivery process.

The proposed second embodiment method addresses the inefficiency limitation in the first embodiment by combining upfront and in-transit batching, enhancing batching probability by enabling bidirectional candidate retrieval, which includes both lookahead (unallocated delivery bookings) and lookback (allocated/dispatched delivery bookings) options when running the batching algorithm. In contrast to the first embodiment, which has a restricted pool of batching candidates, bidirectional candidate retrieval allows the proposed method to consider all candidate types simultaneously in their unbatched state, preventing the formation of suboptimal batches and improving efficiency.

In the first embodiment of Upfront Batching 402, the Candidate Pool 406 of delivery bookings are unallocated 404, 422, so their allocated time is after the current time of allocation, constituting lookahead candidate retrieval. In In-transit Batching 426, the system aims to allocate an unallocated delivery booking 424 to dispatched (allocated) delivery bookings 416 that have not been picked up by its allocated delivery vehicle, meaning the timestamp of allocation will be before the current time of allocation, which is considered lookback candidate retrieval.

In the second embodiment, the bidirectional candidate retrieval method 500 stores both unallocated and allocated (dispatched) delivery bookings in a shared Candidate Pool 504. Unallocated delivery bookings include new unallocated delivery bookings 502 (similar to 404 in the first embodiment), unbatched delivery bookings that have not exceeded the batching pooling time 538 (similar to 422 in the first embodiment) and child delivery bookings 534 from a batch that has been reconfigured and has not been dispatched. Allocated (dispatched) delivery bookings include dispatched solo delivery bookings 540, dispatched batches 536 and child delivery bookings 534 from a batch that has been reconfigured and has not been dispatched, but were previously part of another batch that has been dispatched.

The proposed method 500 allows the unified batching system to consider a larger pool of delivery bookings during batching. This bidirectional candidate retrieval, comprising lookahead and lookback of eligible bookings, offers a more comprehensive and efficient approach to batching.

The unified batching system 500 adopts a recycling and discard strategy, both of which will be further discussed below. The system conducts regular checks 514 and 526 to determine the strategy, recycling or discard strategy, to be employed for each delivery booking.

If checks 514, 526 conclude that the delivery booking(s) in question has/have not been picked up by an allocated delivery vehicle, the system will send the delivery booking(s) for dispatch. For each of the dispatched delivery booking, the system will send an additional copy (i.e., a duplicate) of the dispatched delivery booking in its unbatched state back into a Candidate Pool 504 for batching. These additional copies can stay in the Candidate Pool 504 until they are being picked up by their allocated delivery vehicle. This is known as the recycling strategy.

The purpose of the recycling strategy is to increase the size of the candidate pool by recycling additional copies of dispatched delivery bookings that have not been picked up. By having a larger candidate pool, the algorithm's batching probability can be increased, leading to better efficiency choices based on a wider range of batching combinations.

However, adopting a recycling strategy introduces complexity to the system as these additional copies of the dispatched delivery bookings in the Candidate Pool 504 can be picked up any time by their allocated delivery vehicle.

This is where the discard strategy comes into play. If checks 514, 526 conclude that the delivery booking(s) in question has/have been picked up, the system will discard 516, 532 these delivery booking(s) from the Candidate Pool 504.

The purpose of the dispatch strategy is to ensure that delivery booking(s) that has/have been picked up will not go through any batching attempts by the system and will be removed from the Candidate Pool 504.

More details about the recycling and dispatched strategies will be discussed below.

New unallocated delivery bookings 502 received by the system are sent into the Candidate Pool 504. Delivery bookings from the Candidate Pool 504 are then sent into a Batching Engine 506 for an attempt to form batched delivery bookings.

If the batching attempt is unsuccessful, the system 500 will perform a check 510 on the batching pooling time against the batching pooling time window. Each delivery booking has a batching pooling time window, which is a specified time window that determines the maximum amount of time an unbatched delivery booking can stay in the Candidate Pool 504. If the batching pooling time has not exceeded 538 the specified time window, the unbatched delivery booking 508 will be sent back to the Candidate Pool 504 for another batching attempt. If the batching pooling time has exceeded 512 the specified time window, the system will perform a check 514 on whether the unbatched delivery booking has been picked up by an allocated delivery vehicle.

This check 514 is a node where the recycling and discard strategies are implemented for unbatched delivery bookings.

If the unbatched delivery booking has not been picked up 518, the unbatched delivery booking will be dispatched 520 and allocated to a delivery vehicle as a solo delivery booking for pick up and Drop off 522. At the same time, as part of the recycling strategy, the system also sends 540 an additional copy of the dispatched solo delivery booking back to the Candidate Pool 504 for another attempt of batching.

If the unbatched delivery booking has been picked up 516, the system will implement the discard strategy by discarding any copies of the picked unbatched delivery booking from the Candidate Pool 504 and sending it for Drop off 522.

If the batching attempt is successful, a batch 524 comprising a plurality of delivery bookings will be formed. In this context, each delivery booking within a batch is referred to as a “child delivery booking”. Once batched 524, the system performs a check 526 on whether any of the child delivery bookings in the batch have been picked up by their allocated delivery vehicles.

This check 526 is the node where recycling and discard strategies are implemented for batched delivery bookings.

If all the child delivery bookings in the batch have not been picked up, the system will send 528 the batch to be dispatched 520 and allocated to a delivery vehicle for pick up and Drop off 522. At the same time, as part of the recycling strategy, the system also sends 536 an additional copy of each child delivery bookings of the dispatched batch in their unbatched state back to the Candidate Pool 504 for another attempt of batching.

If the batch contains at least one child delivery booking that has been picked up 530, as part of the discard strategy, the system will discard 532 all the child delivery booking(s) that has/have been picked up from the Candidate Pool 504 and send for Drop off 522. Simultaneously, the system will send 534 any child delivery booking(s) that has/have not been picked up back to the Candidate Pool 504 for another batching attempt.

The batching algorithm processes all of these additional copies 536, 540 of dispatched (allocated) delivery bookings that have not yet been picked up by their allocated delivery vehicles, along with the new unallocated delivery bookings 502, as input for a batching attempt and delivery vehicle allocation.

Each batching attempt occurs periodically. In an exemplary embodiment, the system 500 aggregates and sends all stored delivery bookings from the

Candidate Pool 504 to the Batching Engine 506 for a batching attempt every 10 seconds interval.

An exemplary timeline for a typical existing sequential batching system is illustrated in Figure 6, with a 30 seconds pooling time for each upfront and in-transit batching for simplicity:

1. [11:00:00] Delivery bookings 1 and 2 are created 404 and sent to Candidate Pool 406 for Upfront Batching 402
2. [11:00:30] During Upfront Batching 402, delivery bookings 1 and 2 successfully batched 410 and dispatched 416 for pick up by an allocated delivery vehicle
3. [11:01:00] Delivery bookings 3 and 4 are created 404 and sent to Candidate Pool 406 for Upfront Batching 402
4. [11:01:30] Upfront pooling time for delivery bookings 3 and 4 has exceeded 424. Delivery bookings 3 and 4 fail to be batched 412 during Upfront Batching 402 and proceed to In-transit Batching 426 in unbatched state.
5. [11:02:00] During In-transit Batching 426, delivery booking 3 successfully batched 432 and dispatched 416 with dispatched batch 1+2 that has not been picked up by its allocated delivery vehicle yet, while delivery booking 4 fails to be batched 434 during in-transit batching and remains in an unbatched state.
6. [11:02:30] In-transit pooling time for delivery booking 4 has exceeded 438 and thus, delivery booking 4 is dispatched 416 as a solo delivery booking for pick up by an allocated delivery vehicle. Delivery bookings 5 and 6 are created 404 and sent to Candidate Pool 406 for Upfront Batching 402.
7. [11:03:00] During Upfront Batching 402, delivery bookings 5 and 6 successfully batched 410 and dispatched 416

An exemplary timeline for the proposed unified batching system with bidirectional candidate retrieval method is illustrated in Figures 7 and 8, with a 30 seconds batching pooling time for simplicity. Figure 7 illustrates an exemplary embodiment where no delivery bookings were found to be picked up by their allocated delivery vehicles during checks 514 and 526:

1. [11:00:00] Delivery bookings 1 and 2 are created 502 and sent to Candidate Pool 504 for unified batching 500
2. [11:00:30] During unified batching 500, delivery bookings 1 and 2 successfully batched 524. Each delivery bookings in a batch are referred to as child delivery bookings. The system performs a check 526 if any of child delivery bookings 1 or 2 has/have been picked up in batch 1+2 and finds the answer to be no 528. Batch 1+2 is dispatched 520 for pick up by an allocated delivery vehicle. At the same time, an additional copy of each dispatched child delivery bookings 1 and 2 are being sent back 536 to the Candidate Pool 504 in their unbatched state.
3. [11:01:00] Delivery bookings 3 and 4 are created 502 and sent to Candidate Pool 504 for unified batching 500
4. [11:01:30] During unified batching 500, delivery booking 3 is batched 524 with the copies of child delivery bookings 1 and 2 as batched 1+2+3 has higher efficiency than previously batched 1+2. The system performs a check 526 if any of child delivery bookings 1, 2 or 3 has/have been picked up in batch 1+2+3 and finds the answer to be no 528. Batch 1+2+3 is dispatched 520 for pick up by a newly allocated delivery vehicle. At the same time, an additional copy of each dispatched child delivery bookings 1, 2 and 3 are being sent back 536 to the Candidate Pool 504 in their unbatched state. Batching pooling time for delivery booking 4 has exceeded 512 and delivery booking 4 fails to be batched 508 within the specified batching pooling time. The system performs a check 514 if delivery booking 4 has been picked up and finds the answer to be no 518. Delivery booking 4 is dispatched 520 as a solo delivery booking for pick up by an allocated delivery

vehicle. At the same time, an additional copy of dispatched delivery booking 4 is being sent back 540 to the Candidate Pool 504.

5. [11:02:00] Delivery bookings 5 and 6 are created 502 and sent to Candidate Pool 504 for unified batching 500.
6. [11:02:30] During unified batching 500, delivery bookings 2,3 and 5 are batched 524 as batched 2+3+5 has higher efficiency than previously batched 1+2+3. The system performs a check 526 if any of the child delivery bookings 2, 3 and 5 within batch 2+3+5 has/have been picked up and finds the answer to be no 528. Batch 2+3+5 is dispatched 520 for pick up by a newly allocated delivery vehicle. At the same time, an additional copy of each dispatched child delivery bookings 2, 3 and 5 are being sent back 536 to the Candidate Pool 504 in their unbatched state. Delivery booking 1 becomes unbatched but remains dispatched for pick up by its previously allocated delivery vehicle.. Delivery bookings 4 and 6 are batched 524 during unified batching 500. The system performs a check 526 if any of the child delivery bookings 4 or 6 within batch 4+6 has been picked up and finds the answer to be no 528. Batch 4+5 is dispatched 520 for pick up by a newly allocated delivery vehicle. At the same time, an additional copy of each dispatched child delivery bookings 4 and 6 are being sent back 536 to the Candidate Pool 504 in their unbatched state.
7. [11:03:00] The system performs checks 514, 526 and finds that dispatched delivery bookings 1, 2, 3, 4, 5 and 6 have been picked up by their allocated delivery vehicles. Delivery bookings 1, 2, 3, 4, 5 and 6 are discarded 516, 532 from the Candidate Pool 504.

Figure 8 illustrates an exemplary embodiment where some delivery bookings were found to be picked up by their allocated delivery vehicles during checks 514 and 526:

1. [11:00:00] Delivery bookings 1 and 2 are created 502 and sent to Candidate Pool 504 for unified batching 500

2. [11:00:30] During unified batching 500, delivery bookings 1 and 2 successfully batched 524. The system performs a check 526 if any of child delivery bookings 1 or 2 has been picked up in batch 1+2 and finds the answer to be no 528. Batch 1+2 is dispatched 520 for pick up by an allocated delivery vehicle. At the same time, an additional copy of dispatched child delivery bookings 1 and 2 are being sent back 536 to the Candidate Pool 504 in their unbatched state.
3. [11:01:00] Delivery bookings 3 and 4 are created 502 and sent to Candidate Pool 504 for unified batching 500
4. [11:01:30] During unified batching 500, delivery booking 3 is batched 524 with the copies of child delivery bookings 1 and 2 as batched 1+2+3 has higher efficiency than previously batched 1+2. The system performs a check 526 if any of child delivery bookings 1, 2 or 3 has/have been picked up in batch 1+2+3 and finds the answer to be no 528. Batch 1+2+3 is dispatched 520 for pick up by a newly allocated delivery vehicle. At the same time, an additional copy of dispatched child delivery bookings 1, 2 and 3 are being sent back 536 to the Candidate Pool 504 in their unbatched state. Batching pooling time for delivery booking 4 has exceeded 512 and delivery booking 4 fails to be batched 508 within the specified batching pooling time. The system performs a check 514 if delivery booking 4 has been picked up and finds the answer to be no 518. Delivery booking 4 is dispatched 520 as a solo delivery booking for pick up by an allocated delivery vehicle. At the same time, an additional copy of dispatched delivery booking 4 is sent back 540 to the Candidate Pool 504.
5. [11:02:00] Delivery bookings 5 and 6 are created 502 and sent to Candidate Pool 504 for unified batching 500.
6. [11:02:30] During unified batching 500, delivery bookings 2,3 and 5 are batched 524 as batched 2+3+5 has higher efficiency than previously batched 1+2+3. The system performs a check 526 if any of the child delivery bookings 2, 3 and 5 within batch 2+3+5 has/have been picked up and finds the answer to be yes 530 – child delivery bookings 1, 2

and 3 in batch 1+2+3 have been picked up by their previously allocated delivery vehicle. The system discards 532 child delivery bookings 1, 2 and 3 from Candidate Pool 504 and sends 534 child delivery booking 5 back to Candidate Pool 504. Delivery bookings 4 and 6 are batched 524 during unified batching 500. The system performs a check 526 if any of the child delivery bookings 4 or 6 within batch 4+6 has been picked up and finds the answer to be yes 530 – child delivery booking 4 has been picked up by its previously allocated delivery vehicle. The system discards 532 child delivery booking 4 from Candidate Pool 504 and sends 534 child delivery booking 6 back to Candidate Pool 504 for batching.

7. [11:03:00] Batching pooling time for delivery bookings 5 and 6 has exceeded 512 and delivery bookings 5 and 6 fail to be batched 508 within the specified batching pooling time. The system performs a check 514 if delivery bookings 5 and 6 have been picked up and finds the answer to be no 518. Delivery bookings 5 and 6 are dispatched 520 individually as a solo delivery bookings for pick up by an allocated delivery vehicle each. At the same time, an additional copy of each dispatched delivery bookings 5 and 6 are being sent back 540 to the Candidate Pool 504.

Exemplary embodiments as illustrated in Figures 6-8 can be in the context of, for example, delivery of e-commerce goods ride hailing services or both.

From Figures 6-8, we can see that there are more potential batching combinations available in the proposed unified batching system compared to the existing sequential batching system. Expanding the Candidate Pool of delivery bookings through bidirectional candidate retrieval in the proposed unified batching system increases the algorithms batching probability, allowing for improved batching efficiency across a wider range of batching combinations.

Claims

1. A computer implemented method performed in a communication server apparatus for a delivery platform provider, the method comprising, under control of a processor of the communication server apparatus:
 - receiving a plurality of delivery bookings;
 - storing data for the received delivery bookings in a candidate pool;
 - determining an allocation of a plurality delivery vehicles to one or more of the delivery bookings in the candidate pool;
 - storing data relating to the allocated delivery bookings and unallocated delivery bookings in the candidate pool;
 - redetermining an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool; and
 - dispatching delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.
2. The method according to claim 1, wherein the received delivery bookings are new unallocated delivery bookings.
3. The method according to claim 1, wherein the plurality of delivery bookings are batched and allocated to a single delivery vehicle to form an allocated batch.
4. The method according to claim 3, wherein the redetermined optimised allocation is based on treating all delivery bookings in the candidate pool as unallocated.
5. The method according to any of the preceding claims, wherein the allocated delivery bookings are removed from the candidate pool after being picked up by their allocated delivery vehicles.

6. A computer implemented method performed in a communication server apparatus for a delivery platform provider, the method comprising, under control of a processor of the communication server apparatus:

receiving a plurality of delivery bookings;

storing data relating to the delivery bookings in a candidate pool;

allocating two or more of the delivery bookings in the candidate pool to a plurality of batches and allocating a delivery vehicle for each of the plurality of batches, wherein the allocation is based on optimising the delivery vehicle route for each batch;

storing data relating to the plurality of batches in the candidate pool;

reallocating the delivery bookings in the candidate pool to a further plurality of batches and allocating a delivery vehicle for each of the further plurality of batches.

7. The method according to claim 6, wherein the reallocation is based on treating all delivery bookings in the candidate pool as unbatched.

8. A computer program or computer program product comprising instructions for implementing the method according to any preceding claims.

9. A computer program carrier carrying the computer program or computer program product according to claim 8, wherein the computer program carrier program or computer program product is one of an electronic signal, optical signal, radio signal or computer-readable storage medium.

10. A non-transitory tangible computer-readable storage medium storing the computer program or computer program according to claim 8.

11. A system comprising
a communication server;

at least one user or merchant communication device having an associated user and configured to initiate an unallocated delivery booking which includes a pick up and drop off location;

at least one driver communication device having an associated delivery vehicle and configured to provide delivery vehicle location data; and

communication network equipment configured to establish communication with the communications server, at least one user communication device, and at least one driver communication device;

wherein the communications server comprises at least one processor(s), at least one memory, the server being configured, under control of one or more of the at least one processor(s), to execute instructions stored in one or more of the at least one memory to:

receive a plurality of delivery bookings from the user or merchant communication devices;

store data for the received delivery bookings in a candidate pool;

determine an allocation of a plurality delivery vehicles to one or more of the delivery bookings in the candidate pool;

store data relating to the allocated delivery bookings and unallocated delivery bookings in the candidate pool;

redetermine an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool; and

dispatch delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.

12. A communication server apparatus for a delivery platform provider, the communication server comprising at least one processor, the communication server apparatus being configured, under control of one or more of the at least one processors, to execute instructions, to:

- receive a plurality of delivery bookings;
- store data relating to the received delivery bookings in a candidate pool;
- determine an allocation of a plurality delivery vehicles to one or more of the delivery bookings in the candidate pool;
- store data relating to the allocated delivery bookings and unallocated delivery bookings in the candidate pool;
- redetermine an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool; and
- dispatch delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.

13. A user communication device for a delivery platform provider, the user communication device comprising a processor and a memory, the user communication device being configured, under control of the processor, to execute instructions stored in the memory, to:

- initiate an unallocated delivery booking to a communication server, wherein the unallocated delivery booking includes a pick up and drop off location;
- receive a confirmation from the communication server confirming that a delivery vehicle has been allocated, wherein the allocation of the delivery vehicle is determined by:
 - allocating a plurality delivery vehicles to one or more delivery bookings;
 - storing data relating to allocated delivery bookings and unallocated delivery bookings in a candidate pool;
 - redetermining an optimised allocation of delivery vehicles to the allocated delivery bookings and the unallocated delivery bookings in the candidate pool; and

dispatching delivery vehicles to the allocated delivery bookings and unallocated delivery bookings according to the redetermination.

14. A user communication device, a driver communication device, or a merchant communication device for a delivery platform provider, the user communication device, the driver communication device, or the merchant communication device comprising a processor and a memory, the user communication device, the driver communication device, or the merchant communication device being configured, under control of the processor, to execute instructions stored in the memory, to perform the method according to any of claims 1 to 7.