Operational strategies for a Personal Shopper Service

Alp M. Arslan
Living Analytics Research Centre, Singapore Management University, amuzaffera@smu.edu.sg
Niels Agatz
Rotterdam School of Management, Erasmus University, nagatz@rsm.nl
Mathias Klapp
School of Engineering, Pontificia Universidad Católica de Chile, maklapp@ing.puc.cl

1. Introduction

To compete with online retailers, brick and mortar stores have partnered with Personal Shopper (PS) service providers. Such services act as aggregators receiving online delivery requests for shopping lists made of items available in affiliated stores. Typically, PS services guarantee delivery to the customer within a short lead time, *e.g.*, $L = 90$ minutes. To do so, an automated dispatcher dynamically chooses whether or not to accept each online request. Then, each accepted request is assigned to a *shopper*, who must purchase it at specific stores and deliver it to the customer on time. PS services are popular in the delivery of groceries, since they integrate online shopping with available inventory at grocery stores; Instacart, Postmates and Deliv are some examples of PS services. PS operations also share similar features with online meal delivery (see, for *e.g.*, Reyes et al. (2018), Steever, Karwan, and Murray (2019), Ulmer et al. (2017)), but additionally execute shopping activities *e.g.*, parking, going through the store, collecting and purchasing the items. Furthermore, each PS request may involve collecting items from multiple stores.

To simplify planning, PS service providers may operate a sequential delivery policy, in which shoppers serve one request at a time. However, it might be advantageous to consolidate multiple requests involving common shopping locations in one trip to prorate fixed store shopping times. An additional gain may be achieved by splitting the service of a request involving shopping at multiple stores into separate tasks served by different shoppers.

Request splitting can increase the fleet's time and capacity utilization (*packing benefit*). Figure 1 illustrates an example in which a customer request r demands delivery from stores m_1 and m_2. Two shoppers are located at k_1 and k_2. Travel and shopping times are displayed above the arcs and nodes, respectively. If service must occur by time $t = 6$, then it is infeasible for one shopper to serve the request alone. However, if the request is split into two separate store-tasks, then both

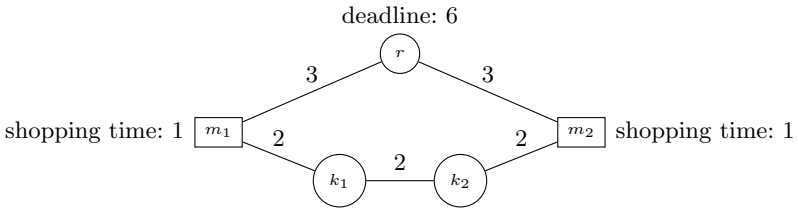


Figure 1 Example of a personal shopper service instance

shoppers can deliver on time. When splitting is allowed, there is also a larger set of routing options (*routing benefit*). Figure 1 shows routing time reductions as a single shopper must travel at least 11 time units to serve r, while two shoppers spend 10 time units. Also, one can save shopping time if multiple tasks originating at one common store are assigned to one shopper (*shopping benefit*).

1.1. Contribution

We introduce the Personal Shopper Problem (PSP), which models the operation of a PS service receiving, accepting and serving same-day delivery requests. Specific contributions are the following. (i) We develop an online policy to efficiently operate a PS service. (ii) We explore three different operational strategies for PS services: serve one request at a time per trip ($1b1$); consolidate multiple services in each trip (C); and split customer requests into store-based tasks and later re-consolidate tasks into each trip ($C\&S$). (iii) We identify three types of potential benefits gained by splitting customer requests: packing, routing and shopping benefits. (iv) We assess our solution quality and confirm the potential benefits of splitting requests in computational experiments.

2. Problem description

Consider an automated dispatcher dynamically accepting requests online throughout the day. Each request r arrives at time e_r and is composed of a set of tasks S_r. Each task $s \in S_r$ requires shopping items at a specific store $m_s \in M$ and, later, delivering it to destination $d_r \in N$ before time $e_r + L$.

When a request is accepted, the dispatcher has to immediately assign its service to one or multiple *shoppers* within an available fleet and communicate a feasible delivery plan covering all pending tasks (a trip per shopper). We assume full knowledge of network travel and shopping times; these last consist in a fixed store shopping time f_m and a time p_m^s to pick up each task s at store m.

The objective of the PSP is to maximize the number of requests served subject to a limited fleet of shoppers. We assume that the dispatcher has no prior nor probabilistic information regarding future requests, so they always accept a request when it is possible to attend it. This approach also avoids discriminatory actions; see Ingold and Soper (2016).

3. Solution

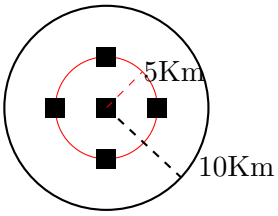
We use an event-based rolling horizon framework commonly used in the online routing literature (Srour, Agatz, and Oppen (2016), Arslan et al. (2019)). Each time a new request arrives, we solve

an instance of a deterministic integer program referred as the *Pickup and Split Delivery Problem with Deadlines* (PsDPd). Each feasible solution to the PsDPd assigns pending tasks to potentially different shoppers and builds shopper trips. So, it defines a valid delivery covering both newly arrived and pending active tasks. If such a plan is found, then we accept the new request, update the delivery plan accordingly and execute it until the next decision epoch. Instead of solving a feasibility problem, we set the objective of the PsDPd to minimize the total duration of all shopper trips to increase shopper's utilization.

Finding an optimal solution to the PsDPd may be time-consuming for real-sized problems. So, we propose a to solve it heuristically. When a request arrives, we try inserting all its tasks into the delivery plan. For each unassigned task, the procedure determines the cheapest insertion positions for its pickup and delivery sub-tasks over all possible options and shopper trips. Then, the cheapest unassigned task is inserted into the plan. This process continues until there is no unassigned task left to insert (or when it is not feasible to do so). Later, we run ALNS over specific neighborhoods to improve the solution. Our heuristic's performance is compared to an exact approach executing smart enumeration over 40 small instances up to 15 tasks. The heuristic finds a feasible solution and, thus, makes a correct acceptance decision in all but one case.

4. Experiments and results

We present a computational experiment to assess our solution's quality. Figure 2 describes the experimental setting and layout of the service area with a 10km radius in which the request destinations are uniformly distributed. We consider five stores in the region (squares). We set a fleet of 3 shoppers and 80 request arrivals uniformly distributed within a ten hour service period. For simplicity, we set that each request requires to shop three homogeneous tasks, each at a different store randomly selected.



Shopper speed	30 km/h
f_m	9 min
p^s_m	1 min
Deadline	90 min
Shopper capacity	10 tasks

Figure 2 **Layout of the service region and parameter setting used in our experiment**

Table 1 presents average results over 99 random realizations of the request arrival process. A substantial performance improvement is obtained by consolidating requests. Comparing the policy C to the $1b1$ policy, the average number of served requests increases from 45.8% to 77.3%. This improvement is associated with a reduction in the fulfillment time per request, *i.e.*, the total time

Table 1 **Average results over all realizations for each operational strategy**

	$1b1$	C	$C\&S$
Served requests (%)	45.8	77.3	88.1
Requests split (%)	0	0	69.2
delivery interval (min.)	0	0	23.6
Fulfillment time per request (min.)	51.2	28.9	25.1
Shopping time per request (min.)	30.0	15.0	10.3
Travel time per request (min.).	21.2	13.9	14.8
# locations visited per request	4.0	2.3	2.7
Click-to-door time (min.)	77.1	78.2	77.3

worked by all shoppers over the number of requests served. We further break down this value in two: shopping and travel time per request. Both improve for policy C compared to $1b1$.

An extra performance increase 77.3% to 88.1% is possible when request splitting is allowed. Also, this improvement exists in all but two realizations. Splitting adds consolidation opportunities by reducing granularity and increasing the *packing benefit*. Moreover, we observe a reduction in the shopping time per request, which indicates extra *shopping benefits*. Conversely, there is a slight increase in travel time per request, *i.e.*, from 2.3 to 2.7. This relates to the fact that a request destination is visited multiple times, but travel time per location visited decreases from 6.0 (13.9/2.3) to 5.4 (14.8/2.7). Finally, we also see that 69.2% of all services are split into multiple shoppers. The average time between the first and last partial delivery, *i.e.*, the delivery interval, is 23.5 minutes while the click-to-door time remains stable over all strategies; this suggests that the first partial deliveries occur earlier than the non-split deliveries.

References

Arslan AM, Agatz N, Kroon L, Zuidwijk R, 2019 *Crowdsourced delivery: A dynamic pickup and delivery problem with ad hoc drivers. Transportation Science* 53(1):222–235.

Ingold D, Soper S, 2016 *Amazon doesn't consider the race of its customers. should it?* URL `https://www.bloomberg.com/graphics/2016-amazon-same-day/`, accessed on 12/12/2019.

Pisinger D, Ropke S, 2010 *Large neighborhood search. Handbook of metaheuristics*, 399–419 (Springer).

Reyes D, Erera A, Savelsbergh M, Sahasrabudhe S, ONeil R, 2018 *The meal delivery routing problem. Optimization Online* .

Srour FJ, Agatz N, Oppen J, 2016 *Strategies for handling temporal uncertainty in pickup and delivery problems with time windows. Transportation Science* 52(1):3–19.

Steever Z, Karwan M, Murray C, 2019 *Dynamic courier routing for a food delivery service. Computers & Operations Research* 107:173–188.

Ulmer MW, Thomas BW, Campbell AM, Woyak N, 2017 *The restaurant meal delivery problem: Dynamic pick-up and delivery with deadlines and random ready times.* Technical report, Technical Report.