The Probabilistic Travelling Salesman Problem with Crowdsourcing

Alberto Santini[1], Ana Viana[2], Xenia Klimentova[3], and João Pedro Pedroso[4]

[1]Universitat Pompeu Fabra, Barcelona, Spain — *alberto.santini@upf.edu*
[2]INESC TEC and Polytechnic of Porto, Portugal — *ana.viana@inesctec.pt*
[3]INESC TEC, Porto, Portugal — *xenia.klimentova@inesctec.pt*
[4]INESC TEC and Universidade do Porto, Portugal — *jpp@fc.up.pt*

March 8, 2021

Abstract

We study a variant of the Probabilistic Travelling Salesman Problem arising when retailers crowdsource last-mile deliveries to their own customers, who can refuse or accept in exchange for a reward. A planner must identify which end-of-day deliveries to offer, knowing that all deliveries need fulfilment, either via crowdsourcing or using the retailer's own vehicle. We formalise the problem and position it in both the literature about crowdsourcing and among routing problems in which not all customers need a visit. We show that to evaluate the objective function of this stochastic problem for even one solution, one needs to solve an exponential number of Travelling Salesman Problems. To address this complexity, we propose Machine Learning and Monte Carlo simulation methods to approximate the objective function, and both a branch-and-bound algorithm and heuristics to reduce the number of evaluations. We show that these approaches work well on small size instances and derive managerial insights on the economic and environmental benefits of crowdsourcing to customers.

Keywords: last-mile delivery; crowdsourcing; stochastic routing.

1 Introduction

With the increasing growth of e-commerce worldwide, business models for last-mile delivery (LMD) of parcels need to innovate and consider fast, cheap and reliable transportation to end customers. One possibility is to crowdsource some deliveries through digital platforms to non-professional couriers who use their own transportation means. While reducing costs and providing a source of income for people who might otherwise be off the labour market [18], crowdsourcing also raises concerns about job quality, environmental impact [30] and trust [22].

The main motivation for this work is to investigate the practice of *crowdsourcing delivery to end customers* as a system that takes advantage of the benefits of crowdsourcing while mitigating the major associated drawbacks. We address the case of retail companies with physical shops that also sell online and are most interested in end-of-day deliveries, i.e., those scheduled around the time when the physical shops close. These constitute a sizeable part of all deliveries (in particular for groceries [48, 49]) because the delivery time coincides with most workers returning home after a workday.

The real-life case prompting this work is, indeed, that of a supermarket chain that offers home delivery of groceries. In traditional LMD, a professional fleet either owned by the chain or outsourced, would deliver the groceries. In current crowdsourced LMD models, the chain would have a platform where potential couriers enrol, get offers for some deliveries, and accept or refuse the offers (this is the model of, e.g., Amazon Flex). Because there is no consolidation of parcels and couriers make journeys that they otherwise would not, this system generates both extra traffic and emissions.

In our case, instead, the chain has a loyalty programme in which the enrolled clients provide their home address. This enables us to defend a slightly different model where *the crowd* is composed of clients that are already in the store and whose home address is close to one of the delivery points. Because the clients are already heading home, we reduce the number of cars associated to the ecosystem supermarket-delivery and reduce the number of miles travelled. Under this model, the planner offers participating clients to deliver someone else's groceries in exchange for a discount. Since clients can accept or refuse the offer, at the end of the day a supermarket vehicle will serve all deliveries which the planner did not *crowdsource*. A similar scheme was first discussed in the work of Archetti, Savelsbergh, and Speranza [7], in which the authors consider that either a professional fleet or occasional drivers can carry out deliveries, but assume that the occasional drivers will always accept offers. In this work we assume that each crowdsourcing fee (i.e., the compensation or discount offered to the customer) is fixed and that the probability that some customer accepts a delivery during the day, provided the decision-maker offers it, is also fixed and known. (We refer the reader to, e.g., the recent work of Yildiz and Savelsbergh [57] discussing optimisation models to set the fees and estimate the probabilities.) We also assume that the supermarket's fleet comprises a single vehicle, because in our motivating example each supermarket manages its own home deliveries in a small geographical area. This assumption is, in any case, easy to relax.

Under the assumptions above, our problem becomes a stochastic generalisation of the Travelling Salesman Problem which we name the **Probabilistic Travelling Salesman Problem with Crowdsourcing (PTSPC)**. Other generalisations of the TSP which do not require to visit all customers have been studied extensively. In particular, as we discuss in Section 2, the literature considers the extreme cases in which the tour planer has either *no power* or *total power* in deciding which clients should not be visited. Our problem sits in between these two extremes.

1.1 Contributions

The main contributions of this paper are the following:

- We introduce the PTSPC, a stochastic generalisation of the TSP. Other well-known routing problems, such as the Probabilistic TSP and the Profitable Tour Problem, are special cases of the PTSPC (see Section 2.2).
- From the point of view of applications, our problem models the case of a company which wants to crowdsource its end-of-day deliveries to its own customers, who can either accept or refuse the offer. From a theoretical point of view, our problem occupies a novel niche in the spectrum of TSP generalisations in which the planner has intermediate power of deciding which customers to visit (see Section 2).
- We formalise this problem and show that computing the objective function of even one solution, requires solving an exponential number of TSPs (each of which is $\mathcal{NP}$-hard). Thus, we devise efficient (exact and heuristic) ways to explore the solution space and to approximate the objective value of the solutions (see Section 4). In particular, in Section 4.3.2 we show how to use Machine Learning techniques to accurately predict the value of the objective function of the PTSPC.
- With an extensive computational analysis, in Section 5 we derive insights on the benefits of crowdsourcing and prove that the algorithms we propose are suitable for integration in a support decision system because they can provide high quality solutions in short times.

2 Problem description

We begin by placing the PTSPC in a spectrum of TSP generalisations which do not require to visit all customers, along a gradient of "decision power" attributed to the tour planner.

At one extreme of this gradient lies the case in which the tour planner has no choice over which customer will require a delivery (we assume we visit customers to deliver some goods) and which will not, because a random variable determines customer presence. This problem is the *Probabilistic Travelling Salesman*

Problem (PTSP) [38]. In the PTSP a decision-maker has to devise a tour to visit a set of delivery points, some of which might be later revealed not to be available. Because the decision-maker does not know in advance which customers will drop out, he/she faces two options.

- The first option, sometimes called the *reoptimisation* approach, is to solve a TSP problem for each possible set of delivery points, wait until the status of all customers is revealed and use the TSP tour visiting the customers requiring delivery. Using strategy is computationally expensive, but it can also be inconvenient for operational reasons; for example, if used over multiple days, it can produce every day a radically different tour while still visiting a similar set of delivery points (see, e.g., [29] for the importance of consistency in parcel delivery).
- A second option, called the *a priori* approach, addresses this concern. It consists in first planning an a priori tour visiting all the customers; when the stochastic outcome is revealed, the decision-maker amends the solution skipping the deliveries that are not required, and performing the remaining ones in the same order as they appear in the a priori tour. This is the first approach introduced, together with the definition of PTSP, by Jaillet [36]. It has the advantage that, when the problem is solved for a multi-day planning horizon, all routes will be similar.

At the opposite extreme there is the case in which the tour planner has total control over which deliveries to perform, giving rise to the *Profitable Tour Problem* (PTP) [21, 24]. In this case, visiting a delivery point earns a profit, while traversing an edge incurs into a cost. The objective is to select the deliveries and plan the tour that maximise the difference between collected profits and travel costs.

Intermediate cases arise when the visit requirements are stochastic, but the decision-maker has some leverage on their outcome. In the PTSPC, for example, the planner has the power of forcing a visit with the retailer's own vehicle if he/she never offers the corresponding delivery for crowdsourcing. One could imagine more complicated interactions in which, e.g., the decision-maker can increase (decrease) the compensation offered to raise (lower) the probability of customers accepting a delivery (see, e.g., [10]).

2.1 Formalisation of the PTSPC

Consider a complete undirected graph $G = (V', E)$ with vertex set $V' = \{0, 1, \ldots, n\}$. Vertex 0 represents the depot, while $V = \{1, \ldots, n\}$ is the set of delivery locations. Let $c_{ij} \in \mathbb{R}^+$ be the cost of traversing an edge $\{i, j\} \in E$ and assume that, if the planner offers delivery $i \in V$ for crowdsourcing, there is a probability $p_i \in [0, 1]$ that some provider will accept the offer. In this case, the decision-maker pays a fee $m_i \in \mathbb{R}^+$ and removes i from the list of customers to visit. Otherwise, if the offer is not accepted, the planner needs to visit i with the retailer's own vehicle. We assume that probabilities p_i are independent which, under our motivating example, is a reasonable approximation: we expect the number of potential couriers to be larger than the number of deliveries, and to offer at most one delivery to each customer. Different assumptions might hold if the planner outsourced deliveries to a logistic provider (e.g., the provider could accept or refuse in block deliveries in a same area). To estimate the values of p_i a practitioner could use, for example, historical data on the success rate of crowdsourcing deliveries to the same area, or an estimation method based on the density of customers living within a small radius from the delivery point.

Denote with $O \subseteq V$ the subset of deliveries offered for crowdsourcing and with $A \subseteq O$ the set of accepted offers, which is only revealed at the end of the day (see Figure 1). The decision-maker has to decide which deliveries to offer for crowdsourcing, i.e., the elements of O, assuming that he/she will reoptimise the end-of-day tour after set A is revealed. In other words, the planner looks for the set O with the lowest expected cost with respect to the random variable A.

This approach can appear similar to the reoptimisation approach for the PTSP, but the two differ in an important aspect. In the PTSP the decision-maker cannot affect the outcome of the random variables: he/she will wait for their realisation and then choose the tour through the customers requiring a visit. In short, the reoptimisation PTSP becomes equivalent to a sequence of TSPs on the operational level. On the tactical level, the PTSP decision-maker can calculate the expected cost of the reoptimised tour (using a weighted average of all the tour costs) but, differently from our problem, cannot act to decrease it.

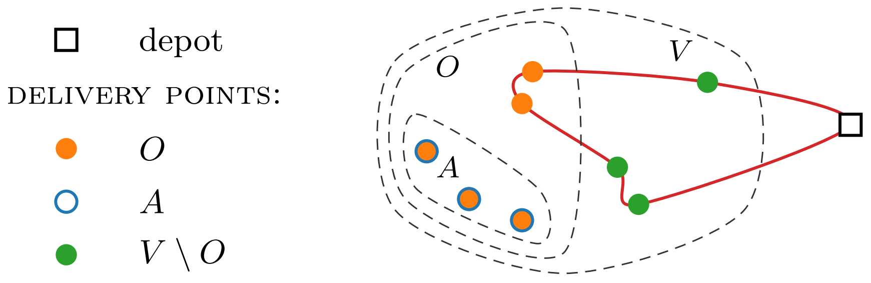

Figure 1: Relation between sets V, O and A. The figure also shows the TSP tour of the owned vehicle when A is the set of deliveries accepted for crowdsourcing.

Let $c_{V'\setminus A}$ be the travel cost of the *reoptimised* tour of the PTSPC: the shortest simple tour starting and ending at the depot, visiting all delivery points which were either not offered, or whose offer for crowdsourcing was not accepted. (Cost c_X, for a generic subset X of vertices is defined formally in Appendix A.) The cost associated with offering deliveries O and having deliveries A accepted is the sum of the crowdsourcing fees for the accepted deliveries plus the cost of the reoptimised tour: $C(O,A) = \sum_{i\in A} m_i + c_{V'\setminus A}$. The probability that a particular set $A \subseteq O$ is the set of accepted deliveries is $\prod_{i\in A} p_i \prod_{i\in O\setminus A}(1-p_i)$. We can then calculate, for a fixed set O of deliveries offered for crowdsourcing, what is the expected cost $\mathbb{E}_A\big[C(O)\big]$ over all possible realisations of A.

$$\mathbb{E}_A\big[C(O)\big] = \sum_{A\subseteq O} \left[\underbrace{\left(\prod_{i\in A} p_i \prod_{i\in O\setminus A} (1-p_i) \right)}_{\text{Prob. that } A \text{ is the accepted set}} \cdot \underbrace{\left(\sum_{i\in A} m_i + c_{V'\setminus A} \right)}_{\substack{C(O,A) \;=\; \text{cost when } A \\ \text{is the accepted set}}} \right] \tag{1}$$

The objective of the problem is to find the set O^{opt} which gives the lowest expected cost:

$$O^{\text{opt}} = \operatorname*{arg\,min}_{O\subseteq V} \; \mathbb{E}_A\big[C(O)\big] \tag{2}$$

Any algorithm which aims to solve (2) faces two challenges. First, the solution space 2^V grows exponentially with the number of delivery points. Second, and differently from most other optimisation problems, the evaluation of the objective function of even one solution is costly: to compute $\mathbb{E}_A\big[C(O)\big]$ one has to solve $2^{|O|}$ TSPs, each of which is $\mathcal{NP}$-hard. Our approach will, thus, focus on tackling both aspects. In Section 4.1 we introduce an exact branch-and-bound algorithm to avoid the complete enumeration of all sets $O \subseteq V$; in Section 4.2 we propose alternative, heuristic, strategies to explore the solution space; in Section 4.3 we propose approximation methods to efficiently estimate the value of $\mathbb{E}_A\big[C(O)\big]$.

2.2 Relationship with classical TSP problems

The PTSPC is a generalisation of three well-studied routing problems. When the crowdsourcing costs are large, $m_i = \infty \ \forall i \in V$, there is no incentive to offer any delivery for crowdsourcing and the planner will decide to perform all deliveries with the retailer's own vehicle. In this case, the PTSPC becomes the classical **Travelling Salesman Problem**. The reduction to the TSP also happens when all probabilities of providers accepting a crowdsourcing offer are zero, $p_i = 0 \ \forall i \in V$. When there are no crowdsourcing fees, $m_i = 0 \ \forall i \in V$, the planner will try to crowdsource all deliveries. At that point, the sole factor determining which deliveries to serve with the retailer's own vehicle and which not, is the stochasticity related to providers' acceptance probabilities p_i. One thus obtains the **Probabilistic Travelling Salesman Problem** in its reoptimisation version. When the customers always accept crowdsourcing offers, $p_i = 1 \ \forall i \in V$, the problem reduces to the **Profitable Tour Problem** with prizes for performing deliveries with the retailer's own vehicle equal to the savings of the corresponding crowdsourcing fees.

We note that these extreme cases are interesting from a theoretical point of view, but hard to happen in practice. For the case study considered it is , thus, important to devise a solution method which works for the general PTSPC because reduction to other problems is unlikely. At the same time, the solution methods we devise for the PTSPC heavily address its specificity (e.g., the complexity of evaluating its objective). Thus, we do not expect our algorithms to be competitive with state-of-the-art methods for classical problems such as the TSP and a practitioner faced with such problems should browse the extensive existing literature on the topic.

3 Literature review

In this section we position the PTSPC in a broader context, highlighting the characteristics it shares with other non-deterministic routing problems and with other optimisation problems arising when integrating crowdshipping in LMD. We also briefly note that the PTSPC (and the PTSP) belong to a growing group of stochastic combinatorial optimisation problems in which the data affected by uncertainty is modelled with Bernoulli random variables (see, e.g., [1, 11, 32, 44]). By contrast, in the majority of problems arising in logistics, stochastic quantities such as travel times [39], costs [55], release dates [7] or demands [13] are modelled with continuous random variables.

3.1 Crowdsourcing in last-mile delivery

In a recent survey, Alnaggar, Gzara, and Bookbinder [2] review operational research literature on crowdsourced LMD and propose a classification of problems arising in the field. Under their classification scheme, the PTSPC: (i) focuses on *e-retailers*, i.e., the company offering the delivery is the same that sells the product; (ii) offers a per-delivery rate determined by the company; (iii) uses pre-planned trips because drivers were already heading in the direction of the delivery points; (iv) focuses on self--scheduling individuals, because the customers enter the supermarket at their own convenience; and (v) considers short-haul deliveries within the same city. These characteristics set the PTSPC apart from most other problems considered in the literature, which focus on crowdsourcing to professional couriers rather than truly occasional drivers.

Archetti, Savelsbergh, and Speranza [7] were among the first to address a problem arising in outsourcing in LMD: the Vehicle Routing Problem (VRP) with Occasional Drivers (ODs). In this problem, the company can decide whether to serve a delivery with a vehicle of its own fleet, or to outsource it for a fixed fee. The planner assigns deliveries to ODs which are already heading towards a destination. For the assignment to take place, the delivery point needs to be close to the driver's destination. The model works under the assumption that ODs always accept requests from the company, provided that they fulfil this “closeness” condition. This assumption is important, as optimal solutions tend to use a high percentage of available drivers. The authors propose a Mixed-Integer Programming (MIP) formulation, but must resort to a multi-start heuristic to tackle instances with more than 25 deliveries. They identify three characteristics affecting the profitability of such a schema: the number of available drivers, their flexibility (how far the delivery point can be from the OD's original destination) and the compensation amount.

Other authors extended the VRP with ODs model to incorporate real-life features such as time windows, multiple and split deliveries [40], transshipment nodes [41], and coordinating ODs on bikes or on foot with a delivery truck from which they relay [34, 37]. The MIP model by Huang and Ardiansyah [34] illustrates well how even deterministic problems can be hard to solve, when they require the interaction of traditional and crowdsourced supply chain segments. The authors, in fact, could solve instances with up to 15 delivery points using the Gurobi solver. They could not solve any instance with 20 and

30 customers and, sometimes, after a 4-hour time limit the solver did not even provide a valid integer solution.

Recent literature also addressed dynamic versions of the problem, in which delivery requests and driver availability become known during the day. Arslan et al. [8] consider a real-time system in which drivers make *trip announcements* and the company can then assign them pickups and deliveries which are compatible with their announced travel (i.e., involving only a small detour) and the recipients' time windows. They tackle the dynamic nature of the problem with a rolling horizon approach, corresponding to a planner who takes decisions at different moments during the day. The decisions consist in matching deliveries to drivers and routing the own fleet for deliveries which were not crowdsourced. With a simulation study the authors show how this approach can produce savings, depending on the time flexibility of the drivers and their willingness to take larger detours. They also conclude that this system is more indicated when all parcels share the same origin, such as a central depot, as this greatly reduces the cost to operate the own fleet.

Dayarian and Savelsbergh [20] study a problem which shares some characteristic of the PTSPC. Their model also addresses the case of using in-store customers as occasional drivers and an own fleet in charge of completing the distribution of parcels. Differently from the PTSPC, they simulate each customer individually; if a customer has a destination compatible with a delivery point, they assume that the customer will accept the delivery. The authors consider two cases: a static case, where all customer visits and deliveries are known in advance, and a dynamic case, where information is only known up to a certain time and the planner reoptimises following a rolling horizon approach. Data on the presence and destination of customers is collected while the customers shop inside the store. For the dynamic case, the authors also propose to base the decisions at each epoch on forecasts of future demand and in-store visits, which they obtain averaging sample historical scenarios. Through a simulation study, the authors determine a trade-off between the savings obtained by reducing the own fleet and the risk introduced when relying on customer availability.

Dahle, Andersson, and Christiansen [19] model occasional drivers as stochastic vehicles possibly appearing at random times during the day (according to a known distribution) in a VRP with time windows. The authors use a two-stage stochastic model in which they plan the routes of the own fleet in the first stage and, after OD appearance times are revealed, they amend the routes and assign deliveries to ODs in the second stage. The authors solve instances with 5, 10, 15 or 20 deliveries, 2 own vehicles, and 2 or 3 ODs. To do so, they use a MIP model and enumerate all possible 2^K scenarios, where K is the number of occasional drivers. The problem proves hard to solve: for example, they cannot find the optimal solution within 2 hours of computation for instances with 10 delivery points and 3 ODs.

Finally, Gdowska, Viana, and Pedroso [26] introduced a multi-vehicle problem for delivery with crowdshipping based on the VRP with ODs [7], which is most similar to our problem. In their work, the authors consider a multi-vehicle fleet of own vehicles and propose an agent-oriented bi-level stochastic model and a local search algorithm for its solution. With computational experiments on instances with 15 deliveries, they show that solutions using crowdsourcing can produce savings, but they cannot assess the accuracy of their heuristic, because they lack exact solutions.

3.2 Probabilistic TSP with profits

The literature on the Probabilistic TSP is vast and its full review is out of the scope of this paper (we refer the reader to classic works [12, 14, 15, 35, 36, 38] and surveys [27, 28]). In the following, we focus on a class of problems which shares common characteristics with the PTSPC: the class of Travelling Salesman Problems with Profits and Stochastic Customers (TSPPSC) introduced by Zhang et al. [58]. This class contains three problems, which mirror the three classic problems in the area of TSP with profits: the Orienteering Problem (OP) which maximises the collected profit under an upper bound on tour duration, the Prize-Collecting TSP (PCTSP) which minimises tour duration under a lower bound on collected profit, and the Profitable Tour Problem (PTP) described in Section 2.

For each of these problems, the authors describe the corresponding version with stochastic customers. Of the three new problems, the one most related to the PTSPC is the PTP with Stochastic Customers (PTPSC). In this problem the decision-maker has to select a subset of delivery points and plan an *a priori* tour only visiting the selected points. When the random outcomes are revealed, the planner amends the tour skipping the points not requiring service. If a delivery is not even included in the *a priori* tour, the delivery point will not be visited and the corresponding profit will not be collected. This is analogous to outsourcing the delivery, paying a fee equal to the lost profit to a provider who is always available to accept requests, highlighting a sort of duality between the proposed problem and our PTSPC. In our problem we have certain delivery points but uncertain outsourcing; in the PTPSC there are uncertain delivery points, but certain outsourcing. Looking at probabilities, the decision-maker of the PTPSC has the possibility of setting $p_i = 1$ (exclude from the tour) for those customers he/she does not select in the *a priori* tour; in contrast, in our problem the decision-maker can set $p_i = 0$ (force in the tour) for those customers he/she does not offer for crowdsourcing.

Zhang, Wang, and Liu [59] propose a genetic algorithm to solve the PTPSC with homogeneous probabilities (i.e., all p_i's are the same). Their analysis is limited to 5 instances with 8, 13, 20, 28 and 50 customers. It shows that, for the two largest instances, the genetic algorithm produces in a few seconds better results than solving the non-linear formulation with a commercial solver running for three hours.

Finally, we remark that of the three problems with profits mentioned above, there are no exact algorithms for their stochastic-customer variants in the literature, while heuristics only exist for the PTPSC [59] and the OP with stochastic customers [5, 45, 58].

4 Algorithms

In this section we propose an exact algorithm based on branch-and-bound, and four heuristic algorithms to explore the solution space of the PTSPC in search for the optimal set of offered deliveries. We also introduce two ways to approximate the objective function of the problem, which we can use instead of exact evaluation to further speed-up the heuristic algorithms.

4.1 A branch-and-bound algorithm

We develop a branch-and-bound (B&B) algorithm in which branching is associated with the inclusion or exclusion of delivery points in the set O of offered deliveries. At each node of the tree, we denote with $O \subseteq V$ the deliveries that the planner will offer for crowdsourcing, with $\bar{O} \subseteq V$ the deliveries which the planner will not offer, and with $F \subseteq V$ the deliveries for which a decision has not been made yet. At the root node of the tree $O = \bar{O} = \emptyset$ and $F = V$, while at leaf nodes the planner has decided the status (offered or not offered) of all deliveries and $F = \emptyset$. Branching in the tree amounts to selecting an offer whose status is not fixed (that is, an offer in F) and creating two children nodes: one in which the delivery is offered and one in which it is not offered.

The effectiveness of a B&B algorithm depends on devising upper and lower bounds for the objective value of the problem at each node, allowing to prune large portions of the tree. If the lower bound of a node is larger than the best upper bound currently available, one can be sure that no solution in the subtree originating at that node will be better than the current best solution. Thus, one can avoid exploring that subtree.

In the rest of this section we describe the lower and upper bounds we use during three exploration. Each node of the tree is determined by the three sets $(O, \bar{O}, F)$ described above; we denote with $z(O, \bar{O}, F)$ the best solution in the subtree originating from node $(O, \bar{O}, F)$.

4.1.1 Lower bound $\underaccent{\tilde}{z}$

A lower bound for $z(O, \bar{O}, F)$ is the following:

$$\underaccent{\tilde}{z}(O, \bar{O}, F) = \sum_{i \in V} m_i - \mathrm{PTP}(\bar{O}, O \cup F), \tag{3}$$

where $\mathrm{PTP}(X, Y)$ denotes the value of the optimal solution of a Profitable Tour Problem over delivery points $X \cup Y$ in which points of the set X are forced to be visited. Intuitively, eq. (3) is stating that one can get a lower bound by being optimistic and assuming that the accepted deliveries will be exactly those (among the offered and the unfixed ones) which give the lowest total cost. We give a formal definition of problem $\mathrm{PTP}(X, Y)$ in Appendix B, while the following theorem proves that eq. (3) defines a lower bound.

Theorem 1. *Quantity $\underaccent{\tilde}{z}(O, \bar{O}, F)$ defined in (3) is a lower bound on the value of the objective function, $z(O, \bar{O}, F)$.*

Proof. Selecting the best set of deliveries to crowdsource, when we assume that (i) the couriers will always accept offered deliveries, and (ii) deliveries in set $\bar{O}$ cannot be crowdsourced, corresponds to finding the lowest-cost tour which visits all vertices of $\bar{O}$ (they must be visited with the retailer's own vehicles), while deciding which vertices of $O \cup F$ to visit.

This decision is taken based on whether it is more convenient to visit vertices in $O \cup F$ with the own vehicle or to pay the crowdsourcing fee. We can model this problem as a PTP in which the profit associated with each vertex of $O \cup F$ is equal to the crowdsourcing fee saved if the retailer's own vehicle visits that vertex. Formally, the equivalence between these two problems can be expressed with the following equality:

$$\min_{A \subseteq O \cup F} \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) = \sum_{i \in V} m_i - \mathrm{PTP}(\bar{O}, O \cup F), \tag{4}$$

where, in the right-hand side, we sum all the crowdsourcing fees in the first term and then we subtract those of the vertices visited by the retailer's own vehicle in the second term (these are the profits of the vertices included in the optimal PTP tour).

With eq. (4), we are ready to show that $\underaccent{\tilde}{z}(O, \bar{O}, F)$ is, indeed, a lower bound for $z(O, \bar{O}, F)$:

$$\begin{aligned}
z(O, \bar{O}, F) = \min_{O' \subseteq F} & \underbrace{\sum_{A \subseteq O \cup O'} \left[\left(\prod_{i \in A} p_i \prod_{i \in (O \cup O') \setminus A} (1 - p_i) \right) \cdot \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) \right]}_{(\star)} \geq \\
& \geq \min_{O' \subseteq F} \min_{A \subseteq O \cup O'} \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) = \\
& = \min_{A \subseteq O \cup F} \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) = \\
& = \sum_{i \in V} m_i - \mathrm{PTP}(\bar{O}, O \cup F) = \\
& = \underaccent{\tilde}{z}(O, \bar{O}, F),
\end{aligned}$$

where the first inequality derives from the fact that

$$\sum_{A \subseteq O \cup O'} \left[\left(\prod_{i \in A} p_i \prod_{i \in (O \cup O') \setminus A} (1 - p_i) \right) \right] = 1$$

and therefore sum $(\star)$ defines a convex combination of the terms $\sum_{i \in A} m_i + c_{V' \setminus A}$. Because the value of a convex combination is always between those of its smallest and largest terms, the first inequality follows. The next equality follows from the fact that $O \cup O' \subseteq O \cup F$ for all $O' \subseteq F$, while the second-last equality is eq. (4) and the last one is the definition of $\underaccent{\tilde}{z}$. $\square$

4.1.2 Upper bound $\bar{z}$

We can trivially define an upper bound for $z(O, \bar{O}, F)$ in the following way:

$$\bar{z}(O, \bar{O}, F) = \sum_{i \in O \cup F} m_i + c_V, \tag{5}$$

where c_V denotes the cost of the TSP tour visiting all delivery locations. The intuition is that in eq. (5) we assume that we will both visit all delivery points with the own vehicle (thus term c_V) and that we will pay all crowdsourcing fees for offered or unfixed deliveries. The following theorem shows that $\bar{z}$ is, indeed, an upper bound for z.

Theorem 2. Quantity $\bar{z}(O, \bar{O}, F)$ defined in (5) is an upper bound on the value of the objective function, $z(O, \bar{O}, F)$.

Proof. The thesis follows from the definition of z:

$$\begin{aligned}
z(O, \bar{O}, F) = \min_{O' \subseteq F} & \underbrace{\sum_{A \subseteq O \cup O'} \left[\left(\prod_{i \in A} p_i \prod_{i \in (O \cup O') \setminus A} (1 - p_i) \right) \cdot \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) \right]}_{(\star)} \leq \\
& \leq \min_{O' \subseteq F} \max_{A \subseteq O \cup O'} \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right) \leq \\
& \leq \min_{O' \subseteq F} \left(\sum_{i \in O \cup O'} m_i + c_V \right) \leq \\
& \leq \sum_{i \in O \cup F} m_i + c_V = \\
& = \bar{z}(O, \bar{O}, F).
\end{aligned}$$

The first inequality is due to $(\star)$ defining a convex combination and, thus, being smaller than its maximal term. The second inequality follows from two other relations. First, $\sum_{i \in A} m_i \leq \sum_{i \in O \cup O'} m_i$ because the sum of non-negative m_i cannot decrease when enlarging the set over which the sum is computed. Second, $c_{V' \setminus A} \leq c_V$ because the cost of the optimal TSP tour cannot decrease when visiting more vertices (assuming, as usual, that the triangle inequality holds). The last inequality is valid because the minimum is smaller than any value its operand takes. The last equality is the definition of $\bar{z}$. □

4.1.3 Upper bound $\bar{z}'$

While $\bar{z}$ is a valid upper bound and it is fast to compute, it is not very tight. We can derive a tighter upper bound $\bar{z}'(O, \bar{O}, F)$ with a reasoning analogous to the one used to derive $\underline{z}$: in the worst case, the realised set A is the one leading to the highest total cost (accounting both the TSP and the crowdsourcing fees). In other words, we take

$$\bar{z}'(O, \bar{O}, F) = \max_{A \subseteq O \cup F} \left\{ \sum_{i \in A} m_i + c_{V' \setminus A} \right\}. \tag{6}$$

The proof that $\bar{z}'$ is a valid lower bound is similar to the proof of Theorem 1, and that $\bar{z}'$ is tighter than $\bar{z}$ follows immediately from its definition. The main issue with eq. (6) is that it requires to solve a hard maximisation problem. We can rewrite this problem as

$$\max_{X \supseteq \bar{O}} \left\{ c_X - \sum_{i \in X} m_i \right\}, \tag{7}$$

which has the same minimiser of (6) (although the two objective functions differ by a constant $\sum_{i \in V} m_i$). The problem defined in eq. (7) is a bi-level integer optimisation problem, in which the inner problem is a TSP, required to compute c_X. The best and, as far as we are aware, only available exact general-purpose solver for bi-level integer optimisation (that of Fischetti et al. [25]) does not support dynamic cut generation schemes such as branch-and-cut. Therefore, we cannot model the inner TSP using an exponential number of subtour-elimination constraints, as in model (27)–(30); we must use a formulation which requires a polynomial number of variables and constraints. In particular, we use the multi-commodity flow reformulation by Sherali, Sarin, and Tsai [53], which is known to have one of the strongest continuous relaxations among compact TSP formulations (see, e.g., Öncan, Altnel, and Laporte [47]). The resulting problem reads as follows:

$$
\begin{aligned}
\max_{\mathbf{x},\mathbf{d},\mathbf{t},\mathbf{y}} \quad & \sum_{i \in V'} \sum_{j \in V'} c_{ij} x_{ij} - \sum_{i \in V} m_i y_i && && (8)\\
\text{s.t.} \quad & \mathbf{x}, \mathbf{d}, \mathbf{t} = \arg\min_{\mathbf{x},\mathbf{d},\mathbf{t}} \sum_{i \in V'_{\mathbf{y}}} \sum_{j \in V'_{\mathbf{y}}} c_{ij} x_{ij} && && (9)\\
& \quad \text{s.t.} \quad \sum_{j \in V'_{\mathbf{y}}} x_{ij} = 1 && \forall i \in V'_{\mathbf{y}} && (10)\\
& \quad \sum_{j \in V'_{\mathbf{y}}} x_{ji} = 1 && \forall j \in V'_{\mathbf{y}} && (11)\\
& \quad d_{ij} + d_{ji} = 1 && \forall i, j \in V_{\mathbf{y}} && (12)\\
& \quad d_{ij} + x_{ij} \leq 2 - d_{jk} - d_{ki} && \forall i, j, k \in V_{\mathbf{y}} && (13)\\
& \quad x_{0i} \leq d_{ij} && \forall i, j \in V_{\mathbf{y}} && (14)\\
& \quad x_{01} \leq d_{ji} && \forall i, j \in V_{\mathbf{y}} && (15)\\
& \quad t^k_{ij} \leq x_{ik} && \forall i, j, k \in V_{\mathbf{y}} \ (i \neq j, k \neq j) && (16)\\
& \quad \sum_{\substack{k \in V_{\mathbf{y}} \\ k \neq j}} t^k_{ij} + x_{ij} = d_{ij} && \forall i, j \in V_{\mathbf{y}} && (17)\\
& \quad \sum_{\substack{i \in V_{\mathbf{y}} \\ i \neq j}} t^k_{ij} + x_{0k} = d_{kj} && \forall k, j \in V_{\mathbf{y}} && (18)\\
& \quad x_{ij} \in \{0, 1\} && \forall i, j \in V'_{\mathbf{y}} && (19)\\
& \quad d_{ij} \in \{0, 1\} && \forall i, j \in V_{\mathbf{y}} && (20)\\
& \quad t^k_{ij} \geq 0 && \forall i, j, k \in V_{\mathbf{y}} && (21)\\
& y_i \in \{0, 1\} && \forall i \in V, && (22)
\end{aligned}
$$

Model (8)–(22) is written for the general case of an asymmetric graph G for ease of notation. Binary variables x_{ij} take value 1 iff arc (i, j) is used in the TSP tour, binary variables d_{ij} take value 1 iff vertex i precedes vertex j in the TSP tour, variables t^k_{ij} arise from a reformulation-linearisation strengthening of subtour-elimination constraints (see [52] and [53]), and binary variables y_i determine whether vertex i should be included in the TSP. We also use notation $V_{\mathbf{y}} = \{i \in V \ : \ y_i = 1\}$ and $V'_{\mathbf{y}} = \{0\} \cup V_{\mathbf{y}}$. We refer the reader to the survey by Öncan, Altnel, and Laporte [47] for a description of each constraint. Because solving model (8)–(22) is time-consuming, we only use bound $\bar{z}'$ at the root node of the B&B tree.

4.1.4 Overall B&B algorithm

Algorithm 1 shows the high-level structure of the B&B algorithm, using a depth-first exploration strategy and branching on the unfixed delivery with the highest crowdsourcing fee. Note that we do not need to recompute the bounds at each node: in the children nodes in which we fix a delivery in O, in fact, the value of the bounds does not change. Furthermore, because the complexity of calculating

Algorithm 1 Branch-an-bound algorithm for the PTSPC.

```
function BranchAndBound
    $O \leftarrow \emptyset$, $\bar{O} \leftarrow \emptyset$, $F \leftarrow V$
    $z^* \leftarrow \infty$  // Cost of the current best feasible solution
    ExploreNode($O, \bar{O}, F,$ true)
    return $z^*$
end function

procedure ExploreNode($O, \bar{O}, F,$ computeBounds)
    if computeBounds then
        $\bar{z} \leftarrow \bar{z}(O, \bar{O}, F)$  // Upper bound $\bar{z}$
        if at root node then
            $\bar{z} \leftarrow \min\{\bar{z}, \bar{z}'(O, \bar{O}, F)$  // Upper bound $\bar{z}'$
        end if
        if $\bar{z} < z^*$ then
            $z^* \leftarrow \bar{z}$  // Update best solution
        end if
        $\underline{z} \leftarrow \underline{z}(O, \bar{O}, F)$  // Lower bound
        if $\underline{z} > z^*$ then
            return  // Prune the tree
        end if
    end if

    if $F = \emptyset$ then  // Leaf node reached
        $z \leftarrow \mathbb{E}_A\big[C(O)\big]$
        if $z < z^*$ then
            $z^* \leftarrow z$  // Update best solution
        end if
        return
    end if

    $i \leftarrow \arg\max_{j \in F} m_j$  // Select delivery to branch on

    ExploreNode($O, \bar{O} \cup \{i\}, F \setminus \{i\},$ true)

    ExploreNode($O \cup \{i\}, \bar{O}, F \setminus \{i\},$ false)
end procedure
```

$\mathbb{E}_A\big[C(O)\big]$ increases with the size of O, it is convenient to explore first the child nodes where deliveries are fixed in $\bar{O}$. In this way we delay the exploration of nodes with large sets O and, possibly, we skip it altogether if the corresponding part of the tree is pruned.

4.1.5 Acceleration strategies

An important property of $\mathbb{E}_A\big[C(O)\big]$ as defined in eq. (1), is that truncating the sum at any point gives a valid lower bound. Thus, if at any moment during the computation of $\mathbb{E}_A\big[C(O)\big]$ the partial sum exceeds the current best solution cost z^*, we can discard the leaf node.

Another way to speed up the algorithm is to use the following bounding technique. Assume we store the values of $\mathbb{E}_A\big[C(O)\big]$ we computed during the exploration of the B&B tree and that we are to compute $\mathbb{E}_A\big[C(O \cup \{j\})\big]$. We can first compute an upper bound without solving any TSP:

$$\mathbb{E}_A\big[C(O \cup \{j\})\big] \leq \mathbb{E}_A\big[C(O)\big] + 2p_j m_j \sum_{A \subseteq O} p_{A,O}\mu_{A,j}, \tag{23}$$

where $p_{A,O} = \prod_{i \in A} p_i \cdot \prod_{i \in O \setminus A}(1 - p_i)$ is the probability that A is the accepted set when O is the offered set, and $\mu_{A,j} = \min_{i \in V \setminus A} c_{ij}$. If the upper bound, which we denote as $\bar{z}_{\text{leaf}}$, is already better than the best solution z^* we can update it as $z^* \leftarrow \bar{z}_{\text{leaf}}$ and we flip a flag indicating that z^* is not the objective value of an actual solution. Only if, when exploring a future node, we find a new solution improving on such a z^*, then we have to calculate the actual value of $\mathbb{E}_A\big[C(O \cup \{j\})\big]$. A parallel implementation could dedicate a thread to compute this value and use the bound while exploring the rest of the tree. The following theorem proves the validity of bound (23).

Theorem 3. Given a set $O \subset V$ and a delivery location $j \in V \setminus O$, inequality (23) is valid.

Proof. For notation convenience we define $m_A = \sum_{i \in A} m_i$. Then, the following equality holds:

$$\begin{aligned}\mathbb{E}_A\big[C(O \cup \{j\})\big] = &\sum_{A \subseteq O} \left[p_{O,A} \cdot p_j \cdot \big(m_A + m_j + c_{(V' \setminus A) \setminus \{j\}}\big) \right] + \\ &\sum_{A \subseteq O} \left[p_{O,A} \cdot (1 - p_j) \cdot \big(m_A + c_{V' \setminus A}\big) \right],\end{aligned}$$

where the first sum considers the subsets of $O \cup \{j\}$ which contain j and the second sum considers those which do not contain j. Denote with $\gamma_{A,j}$ the difference between the costs of the TSPs over $(V' \setminus A) \setminus \{j\}$ and over $V' \setminus A$:

$$\gamma_{A,j} = c_{(V' \setminus A) \setminus \{j\}} - c_{V' \setminus A}.$$

Then we can write

$$\begin{aligned}\mathbb{E}_A\big[C(O \cup \{j\})\big] &= p_j \sum_{A \subseteq O} p_{O,A}\big(m_A + c_{V' \setminus A} + m_j + \gamma_{A,j}\big) + (1 - p_j) \sum_{A \subseteq O} p_{O,A}\big(m_a + c_{V' \setminus A}\big) = \\ &= p_j \mathbb{E}_A\big[C(O)\big] + p_j \sum_{A \subseteq O} p_{A,O}\big(m_j + \gamma_{A,j}\big) + (1 - p_j)\mathbb{E}_A\big[C(O)\big] = \\ &= \mathbb{E}_A\big[C(O)\big] + p_j m_j \sum_{A \subseteq O} p_{A,O}\gamma_{A,j}.\end{aligned}$$

To get (23) we bound term $\gamma_{A,j}$ as $\gamma_{A,j} \leq 2 \min_{i \in V' \setminus A} c_{ij}$, which derives from the trivial bound on the cost increase of a TSP when adding a new vertex to visit. $\square$

4.1.6 Fast solution of many similar TSPs

Throughout the exploration of the B&B tree, we must solve many TSPs on subgraphs induced by various subsets of V'. For example, at each leaf node we compute $\mathbb{E}_A\big[C(O)\big]$ which involves solving one TSP over vertices $V' \setminus A$ for each $A \subseteq O$. Clearly one can cache the values $c_{V' \setminus A}$ required to compute $\mathbb{E}_A\big[C(O_1)\big]$ (for a given set $O_1 \subseteq V$) and then re-use the common ones to compute $\mathbb{E}_A\big[C(O_2)\big]$ (for a different set $O_2 \subseteq V$). However, it seems reasonable that even to evaluate one objective value $\mathbb{E}_A\big[C(O)\big]$, parts of the solution needed to obtain $c_{V' \setminus A_1}$ (for some $A_1 \subseteq O$) could be reused to compute $c_{V' \setminus A_2}$ (for some other $A_2 \subseteq O$): intuitively, TSP tours over similar subsets are likely to have subpaths in common.

For this reason, we evaluate two strategies to solve the multiple TSPs required to compute $\mathbb{E}_A\big[C(O)\big]$:

- The first strategy (Concorde) is to disregard the fact that the multiple TSPs over sets $V' \setminus A$ are related and simply solve each of them independently. To this end, we use the popular TSP solver Concorde [6]. It has been shown empirically by Mu et al. [46] that Concorde solve times scale as $a \cdot b^{\sqrt{n}}$, where a and b are constants (with $b \approx 1.25$) and n is the number of vertices. For problems of the size considered in this work, this corresponds to an increase of a factor of approximately $1.25^{\sqrt{21}}/1.25^{\sqrt{4}} \approx 1.78$ between the solution time of the smallest non-trivial TSP (with 4 vertices) and the largest one (with 21 vertices, the size of the largest instance — see Section 5.1). As such, solving one TSP with Concorde is *almost* constant-time in our situation and, therefore, computing $\mathbb{E}_A\big[C(O)\big]$ is *almost* exponential-time, because it requires to solve $2^{|O|}$ TSPs.

- The second strategy (HeldKarp) is to use the classical Held-Karp Dynamic Programming (DP) algorithm [31] to compute $c_{V'}$. The DP algorithm is based on the following cost function: let $X \subseteq V$ and denote with $H(X,t)$ the cost of the shortest Hamiltonian path from 0 to t in the subgraph induced by vertices $\{0,t\} \cup X$. The cost of the optimal TSP tour over V' is then $c'_V = H(V,0)$ (recall that $V = V' \setminus \{0\}$) and the DP recursion used is:

$$H(\emptyset,t) = c_{0t} \tag{24}$$
$$H(X,t) = \min_{i \in X} \big\{H(X \setminus \{i\}, i) + c_{it}\big\}. \tag{25}$$

  The DP table to compute $H(V,0)$ has one column for each subset $X \subseteq V$ and one row for each vertex $i \in V'$; the entry in the column indexed by X and the row indexed by i is $H(V,i)$. Held and Karp's algorithm is not cheap to execute: its time grows as $O(2^n n^2)$ and its space (i.e., the size of the DP table) as $O(2^n n)$. However, once the DP table is build, one can compute c_X for any $\{0\} \subset X \subseteq V'$ in constant time, simply accessing the entry in the column indexed by X and the row indexed by 0.

In a preliminary experiment, we used 3168 instances (see Section 5.1) with 8 to 16 delivery points, i.e., 9 to 17 TSP vertices. For each of them, we used strategies Concorde and HeldKarp to compute c_X for all subsets X of delivery points. The results, summarised in Figure 2, suggest that using strategy HeldKarp is less computationally expensive but much more memory hungry than strategy Concorde, to the point of being impractical to use for larger instances. For this reason, we resort to using strategy Concorde in the computational experiments of Section 5.

4.2 Heuristic algorithms

We propose four heuristic strategies to explore the space of sets $O \subseteq V$, inspired by stepwise and bidirectional heuristic for variable selection in statistics (see, e.g., Hocking [33]):

- The *Forward Stepwise* heuristic (F-Step) starts with $O = \emptyset$ and, at each iteration, adds to O the delivery location which decreases the expected cost the most, if any; otherwise, the heuristic stops.

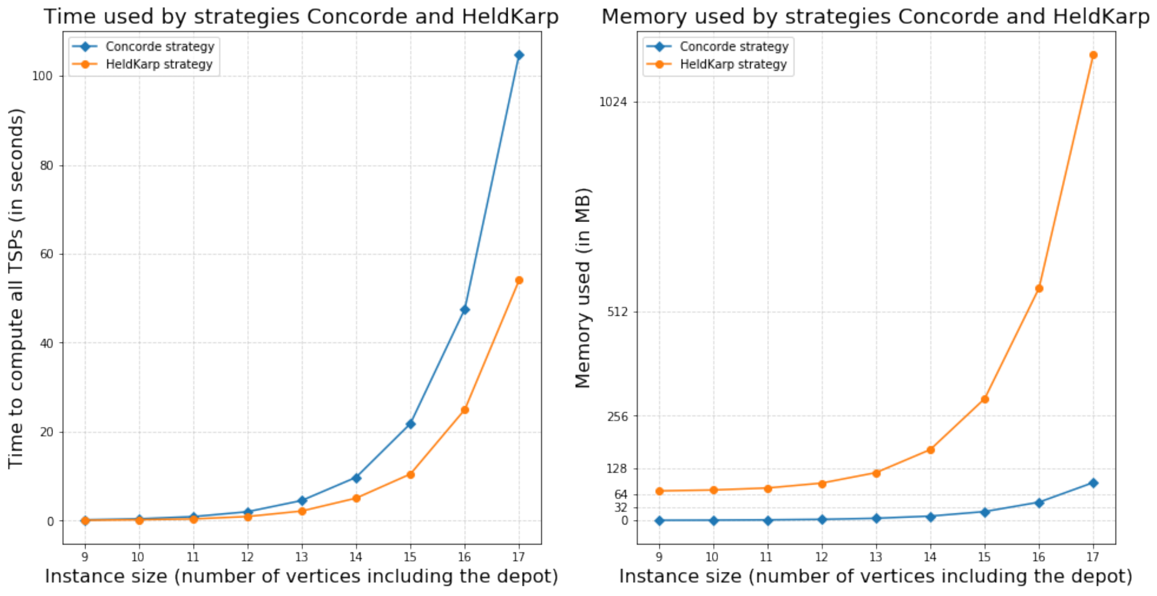

Figure 2: Comparison of strategies CONCORDE and HELDKARP. Each point refers to the time (left) and memory (right) used to compute the cost of all TSPs, averaged over all instances of the same size.

- The *Backward Stepwise* heuristic (B-Step) is analogous, but starts with $O = V$ and, at each iteration removes from O the delivery which decreases the cost the most.
- The *Forward-Backward Bidirectional* heuristic (FB-Bid) starts with $O = \emptyset$ and alternates one forward and one backward phase at each iteration.
- The *Backward-Forward Bidirectional* heuristic (BF-Bid) starts with $O = V$ and alternates one backward and one forward phase.

Algorithm 2 describes in detail the implementation of the FB-Bid heuristic. The other heuristics are implemented similarly.

During preliminary experiments, we implemented variations of the above heuristics which start from random sets $O \subseteq V$. Such variations did not produce any sensible improvement over the more basic versions, so we decided to keep the latter for simplicity.

4.3 Approximation of $\mathbb{E}_A\big[C(O)\big]$

As noted before, computing the objective value $\mathbb{E}_A\big[C(O)\big]$ of a solution O is hard, because one has to evaluate function $C(O, A)$ for each set $A \subseteq O$ (i.e., $2^{|O|}$ times) and solve a TSP at each evaluation. Our hypothesis, however, is that it is possible to approximate $\mathbb{E}_A\big[C(O)\big]$ well, while evaluating much fewer functions. This hypothesis is intuitively motivated by the existence of strong concentration inequalities for both the sum of independent Bernoulli random variables [43, Ch. 4] and the length of the Euclidean TSP tour of points taken uniformly at random in a square [54, Ch. 2]. Because A is, indeed, a realisation of independent Bernoullis and $c_{V' \setminus A}$ is the length of a TSP over points defined by the realisation of A, we can expect $C(O, A)$ to be concentrated around $\mathbb{E}_A\big[C(O)\big]$. The above argument is not formal (e.g., the points in $V \setminus A$ are not chosen uniformly at random) but it motivates the use of methods which work best when the quantity to estimate is concentrated around its mean.

Algorithm 2 The FB-Bid heuristic for the PTSPC.

```
1: procedure CheckImprovingFwd($O, z, \ell$)
2:     $j \leftarrow$ Null

3:     for $i \in V \setminus O \setminus \{\ell\}$ do
4:         if $\mathbb{E}_A\big[C(O \cup \{i\}\big] < z$ then
5:             $z \leftarrow \mathbb{E}_A\big[C(O \cup \{i\}\big]$
6:             $j \leftarrow i$
7:         end if
8:     end for

9:     return j
10: end procedure

11: procedure CheckImprovingBck($O, z, \ell$)
12:     $j \leftarrow$ Null

13:     for $i \in O \setminus \{\ell\}$ do
14:         if $\mathbb{E}_A\big[C(O \setminus \{i\}\big] < z$ then
15:             $z \leftarrow \mathbb{E}_A\big[C(O \setminus \{i\}\big]$
16:             $j \leftarrow i$
17:         end if
18:     end for

19:     return j
20: end procedure

21: function FB-Bid
22:     $O \leftarrow \emptyset$  // Current best set
23:     $z \leftarrow \mathbb{E}_A\big[C(O)\big]$  // Current best cost
24:     $j^{\mathrm{F}} \leftarrow$ Null  // Best customer to add to $O$
25:     $j^{\mathrm{B}} \leftarrow$ Null  // Best customer to remove from $O$

26:     while true do
27:         $j^{\mathrm{F}} \leftarrow$ CheckImprovingFwd($O, z, j^{\mathrm{B}}$)
28:         if $j^{\mathrm{F}} \neq$ Null then
29:             $O \leftarrow O \cup j^{\mathrm{F}}$
30:             $z \leftarrow \mathbb{E}_A\big[C(O)\big]$
31:         end if

32:         $j^{\mathrm{B}} \leftarrow$ CheckImprovingBck($O, z, j^{\mathrm{F}}$)
33:         if $j^{\mathrm{B}} \neq$ Null then
34:             $O \leftarrow O \setminus j^{\mathrm{B}}$
35:             $z \leftarrow \mathbb{E}_A\big[C(O)\big]$
36:         end if

37:         if $j^{\mathrm{F}} =$ Null and $j^{\mathrm{B}} =$ Null then
38:             break
39:         end if
40:     end while

41:     return O
42: end function
```

4.3.1 Monte Carlo simulation

The first such method is *Monte Carlo* (MC) simulation: given a restricted family $\mathcal{A} \subset \mathcal{P}(O)$ of subsets of O, we define the MC estimate of $\mathbb{E}_A\big[C(O)\big]$ as:

$$\hat{\mathbb{E}}_A^{\mathrm{MC}}\big[C(O)\big] = \frac{1}{|\mathcal{A}|} \sum_{A \in \mathcal{A}} \left(\sum_{i \in A} m_i + c_{V' \setminus A} \right), \tag{26}$$

where each set $A \in \mathcal{A}$ is built selecting each point $i \in O$ with probability p_i. Controlling the size of $\mathcal{A}$, a user can trade-off computation time and approximation accuracy. Preliminary experiments showed that already setting $|\mathcal{A}| = 20$ was enough to provide an accuracy level which allows to identify sets O whose expected cost lies well under a 1% gap from the expected cost of the optimal set O^{opt}. We refer the reader to, e.g., Montemanni et al. [45] for more advanced methods of tuning parameter $|\mathcal{A}|$ for the related Probabilistic Orienteering Problem.

4.3.2 Machine Learning

Another valid observation is that calculating $\mathbb{E}_A\big[C(O)\big]$ is faster for small sets O, because there are fewer sets $A \subseteq O$. Defining some size-independent features of O, then, we propose to apply a *Machine Learning* (ML) algorithm to learn $\mathbb{E}_A\big[C(O)\big]$ from a training set of small sets O and then predict it for larger sets.

We first describe the features we use as independent variables. To do so, we must introduce some notation. Let $\bar{c}(W), \hat{c}(W), \underline{c}(W)$ be, respectively, the largest, the average, and the smallest distance of a delivery point in set $W \subseteq V$ from the depot:

$$\bar{c}(W) = \max_{i \in W} c_{0i}, \quad \hat{c}(W) = \frac{1}{|W|} \sum_{i \in W} c_{0i}, \quad \underline{c}(W) = \min_{i \in W} c_{0i}$$

Let $m(W)$ be the sum of crowdsourcing fees for deliveries in W, $m(W) = \sum_{i \in W} m_i$. Finally, let $d(W)$ be the diameter of W, $d(W) = \max_{i,j \in W} c_{ij}$. We define, then, the following features:

- One binary feature for each delivery point $i \in V$, with value 1 iff $i \in O$.
- One feature representing the fraction of crowdsourcing fees of offered deliveries, $m(O)/m(V)$.
- Three features, representing the ratios between largest, average, and smallest distances from the depot, between delivery points in O and all delivery points: $\frac{\bar{c}(O)}{\bar{c}(V)}, \frac{\hat{c}(O)}{\hat{c}(V)}, \frac{\underline{c}(O)}{\underline{c}(V)}$.
- Three features, similar to those above, but referring to $V \setminus O$: $\frac{\bar{c}(V \setminus O)}{\bar{c}(V)}, \frac{\hat{c}(V \setminus O)}{\hat{c}(V)}, \frac{\underline{c}(V \setminus O)}{\underline{c}(V)}$.
- Two features, representing the ratio between the diameters of, respectively, O and $V \setminus O$, and the diameter of V: $\frac{d(O)}{d(V)}, \frac{d(V \setminus O)}{d(V)}$.

We decided to use the features above after data exploration and preliminary experimentation. One can further reduce their number by performing feature selection. However, the feature we use are quick to compute when building the training set and leaving any of them out neither decreases training time nor increases the model's accuracy significantly.

We test five simple and fast-to-train ML models:

- the *Elastic Net* [60];
- a single *Regression Tree* [17];
- a modified regression tree, known as the *M5* model [50];
- a *Random Forest* of regression trees [16];

- an ensemble of regression trees trained with the *AdaBoost.R2* algorithm [23].

To create the training and test sets, we evaluate $\mathbb{E}_A\big[C(O)\big]$ for all sets $O \subseteq V$ on 1250 instances of size 8 to 12 (see Section 5.1). We use the two-thirds smallest sets O for the training set, and the remaining one-third for the test set (we denote the test set as $\mathcal{T}$ in the following).

Let $\hat{\mathbb{E}}_A^{\mathrm{ML}}\big[C(O)\big]$ denote the prediction of an ML algorithm for the expected cost of set O. We use two metrics to assess the accuracy of the models. The first is a classical measure of prediction accuracy for regression models, the mean absolute relative error (MARE):

$$\mathrm{MARE} = 100 \cdot \frac{1}{|\mathcal{T}|} \sum_{O \in \mathcal{T}} \frac{\big|\hat{\mathbb{E}}_A^{\mathrm{ML}}\big[C(O)\big] - \mathbb{E}_A\big[C(O)\big]\big|}{\mathbb{E}_A\big[C(O)\big]}.$$

The second is a metric of interest when the prediction from the ML model is the objective function of an optimisation model: the error on the best set (EB). This is the relative difference between the cost of the set that the ML model identifies as the lowest-cost set in $\mathcal{T}$ and the cost of the actual best set:

$$\hat{O}^{\mathrm{ML}} = \min_{O \in \mathcal{T}} \hat{\mathbb{E}}_A^{\mathrm{ML}}\big[C(O)\big], \quad O^{\mathrm{opt}} = \min_{O \in \mathcal{T}} \mathbb{E}_A\big[C(O)\big], \quad \mathrm{EB} = 100 \cdot \frac{\big|\mathbb{E}_A\big[C(\hat{O}^{\mathrm{ML}})\big] - \mathbb{E}_A\big[C(O^{\mathrm{opt}})\big]\big|}{\mathbb{E}_A\big[C(O^{\mathrm{opt}})\big]}.$$

This metric is important because it measures the relative loss that a planner would incur if he/she used $\hat{O}^{\mathrm{ML}}$ instead of O^{opt} as the offered set.

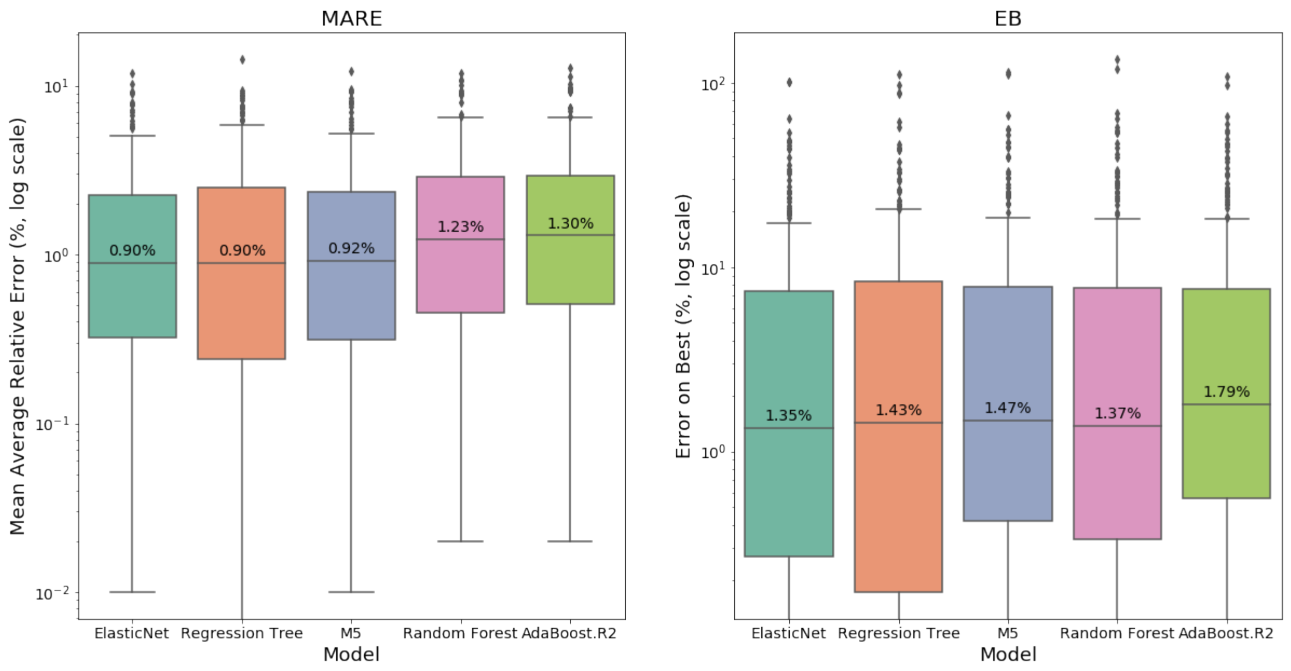

Figure 3: Comparison of MARE and EB error metrics for the five considered Machine Learning model, over the test sets relative to the 1250 instances considered. Each box spans between the first and third quartiles, with the horizontal line and the numbers indicating the median. The rest of the distribution is included between the whiskers, except for outliers which are marked with fliers.

Figure 3 reports the distribution of MARE and EB over the 1250 test sets used in this analysis. The median value for each model is reported in each box, and visualised with a horizontal line; the rest of the box spans between the first and third quartiles. Whiskers extend to the rest of the distribution, except for outliers marked with fliers. Both metrics agree in selecting *Elastic Net* as the best model out of the five we compare. We also note that all methods perform generally well, with the central quartiles of both distributions well below value 10%. As a result of this preliminary computational analysis, we decided to use the *Elastic Net* model.

5 Computational study

After describing the instance generation method we used, we propose two main analyses of results. The first aims at understanding how market environment factors, such as the willingness of customers to accept offers or the crowdsourcing fee amounts, affect planner's profitability and environmental sustainability. The indicator of profitability we use are the savings that the planner achieves when allowing crowdsourcing vs. when serving all deliveries with the retailer's own vehicle. (The cost incurred when serving all deliveries with the own vehicle is the cost of the optimal TSP tour over all delivery points.) The indicator of sustainability is the number of miles saved by the retailer's own vehicle when crowdsourcing. This metric assumes that customers who accept to perform deliveries and use a carbon-emitting mean of transport, only apply a minimal detour to their originally planned routes.

The second analysis focuses on the computational contribution of this paper. We test the performance of the exact and heuristic methods introduced, propose to speed-up one of the heuristic methods using Monte Carlo and Machine Learning objective function estimation, and advise on which algorithms are more appropriate if the decision must take place within a few minutes.

5.1 Instance generation

We generate a large set of synthetic instances to analyse the impact of probabilities p and crowdsourcing fees m on the solutions. We consider instances with $n = 8$ up to $n = 20$ deliveries. In each of them, we place the depot at the origin of the euclidean plane and distribute the delivery points uniformly within a radius $R = 100$ from the depot. In other words, for each delivery point, we generate a radius $r \in [0, R]$, an angle $\theta \in [0, 2\pi)$, and then the coordinates of the point as $x = \sqrt{r}\cos\theta, y = \sqrt{r}\sin\theta$.

We use three criteria to assign probabilities to the delivery points:

- *Uniform* criterion: we first fix a base probability p and then draw each p_i uniformly at random in $(p - 0.05, p + 0.05)$. We consider values of p in $\{0.05, 0.10, \dots, 0.55, 0.6\}$; during preliminary experiments we noticed that higher values lead to solutions in which, for the considered fee values, it is consistently convenient to offer all deliveries for crowdsourcing.
- *Direct proportional* criterion (farther deliveries are easier to crowdsource): we fix a base probability $p \in \{0.25, 0.5\}$ and then let each $p_i = p + \gamma_i \cdot 0.25$, where $\gamma_i \in [-1, +1]$ varies proportionally with the distance between delivery point i and the depot. For a point i at distance 0 from the depot, γ_i would be -1; for a point at distance 100, it would be $+1$.
- *Inverse proportional* criterion (closer deliveries are easier to crowdsource): analogous to the previous one, but γ_i varies inversely proportionally with the distance between the delivery point and the depot.

We use two criteria to assign the crowdsourcing fees:

- *Direct proportional* criterion, i.e., each m_i takes a value in $(0, 100 \cdot \frac{m}{n})$ proportional to the value of p_i compared to the minimum and maximum probabilities assigned to any delivery point. Here n is the number of delivery points and $m \in \{2.5, 2.6, \dots, 3.4, 3.5\}$ is a base fee parameter determining the fee paid to the customers.
- *Inverse proportional* criterion: analogous to the previous one, but the fee is inversely proportional to the probability.

Varying the criteria above we created a dataset of 4576 instances, available on GitHub [51].

5.2 Analysis of the solutions

With this analysis we want to understand how instance generation parameters influence the optimal solutions to the PTSPC. These parameters are linked to properties of real-life scenarios; thus, analysing their impact can lead to managerial insights on which characteristics make crowdsourcing to customers more attractive. For example, if the market for crowdsourced LMD is offer-driven, one can imagine that higher fees would lead to higher crowdsourcing probabilities, a scenario modelled with the *direct proportional* fee criterion. If the market is demand-driven, the probability of crowdsourcing a delivery depends mainly on intrinsic characteristics (e.g., it is low for out-of-the-way delivery points) and a planner must offer high rewards for deliveries which are hard to crowdsource; this scenario is represented by the *inverse proportional* fee criterion. Analogously, one can use the *direct proportional* criterion for probabilities to model a supermarket which is mainly visited by customers living far away and driving a car, whereas the *inverse proportional* can model a supermarket in the city centre, with most of the customers walking there.

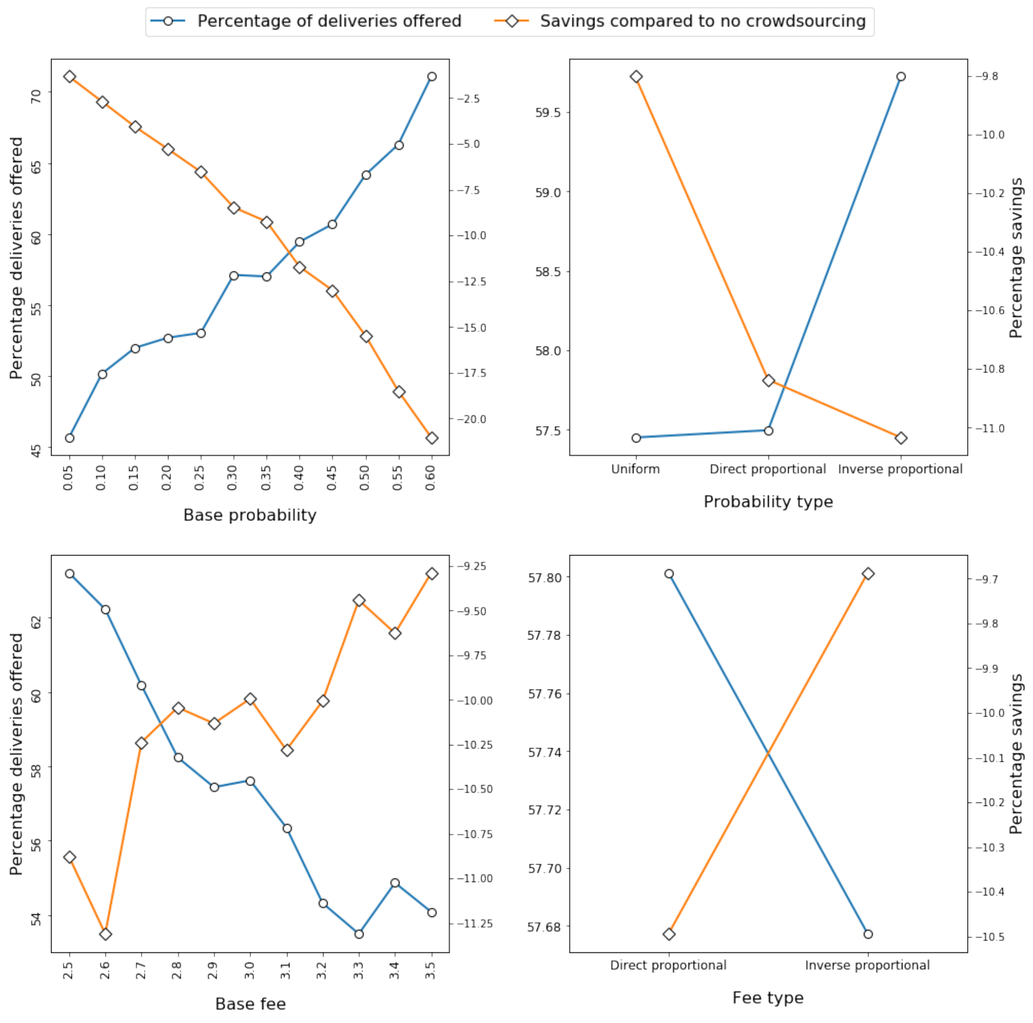

Figure 4: Percentage of deliveries offered and percentage savings compared to no crowdsourcing (i.e., using the retailer's own vehicle for all deliveries), as a function of the instance generation parameters.

Figure 4 shows how two metrics vary with the instance generation parameters. Such parameters are the *base probability* p, the *base fee* m, and the criteria used to generate the other probabilities and fees (see Section 5.1). The first metric is the percentage of deliveries offered, $100 \cdot \frac{|O^{\text{opt}}|}{|V|}$ (reported on the left y axis); the second is the percentage savings when using crowdsourcing, $100 \cdot \frac{\mathbb{E}_A\left[C(O^{\text{opt}})\right] - c_{V'}}{c_{V'}}$ (reported on the right y axis). The figure reports the average of such metrics, over all instances which share the same instance generation parameters.

As expected, increasing the base probability has a large impact on the solution: it yields larger offered sets and increased savings. Increasing the crowdsourcing fees, instead, has the opposite effect. Note, however, how probabilities tend to have a higher discriminative power on the savings achieved. For example, when grouping instances by base probabilities, the instances corresponding to $p = 0.6$ give, on average, more than 20% of savings. If we group instances by base fees, though, even the instances with the lowest fees $m \in \{2.5, 2.6\}$ give a more heterogeneous array of savings, which only averages to circa 11% of savings.

In demand-driven markets the price elasticity curve is often modelled as a sigmoid function (see, e.g., [9]). Because, as noticed above, probabilities of acceptance have a large impact on savings, a planner should price the crowdsourcing fees to maximise the corresponding increase in probabilities (i.e., up until the elasticity curve starts flattening). After that, one can conceive alternative methods of improving the probability of acceptance without further increasing the fees; e.g., via marketing or providing alternative benefits. The specific relation between an increase in the fee and the corresponding increase in acceptance probability is market-specific and, to some extent, even customer-specific because different people have different price sensitivities. Therefore, although the planner should estimate this relation based on empirical data, as a rule of thumb we notice that the same relative variation applied to probabilities has a larger effect than applied to fees. For example, increasing the base probabilities by +40% (from 0.25 to 0.35) causes an increase in savings of 42% (from 6.53% to 9.25%). But a similar +40% increase in the base fees (from 2.5 to 3.5) only decreases the savings by 15% (from 10.88% to 9.29%). As such, if the relationship between offered fee and acceptance probability were roughly linear in this interval, we would have a net +27% increase in savings under the high-fee/high-probability scenario compared to the low-fee/low-probability one. Because the critical part of the sigmoid elasticity curve is approximately linear, such an analysis seems plausible.

The two considered metrics are also stable when aggregating by the probability type and fee type parameters, with variations within 1–2 percentage points. This suggests that the advantages a planner can get by crowdsourcing last-mile deliveries are robust over the scenarios considered.

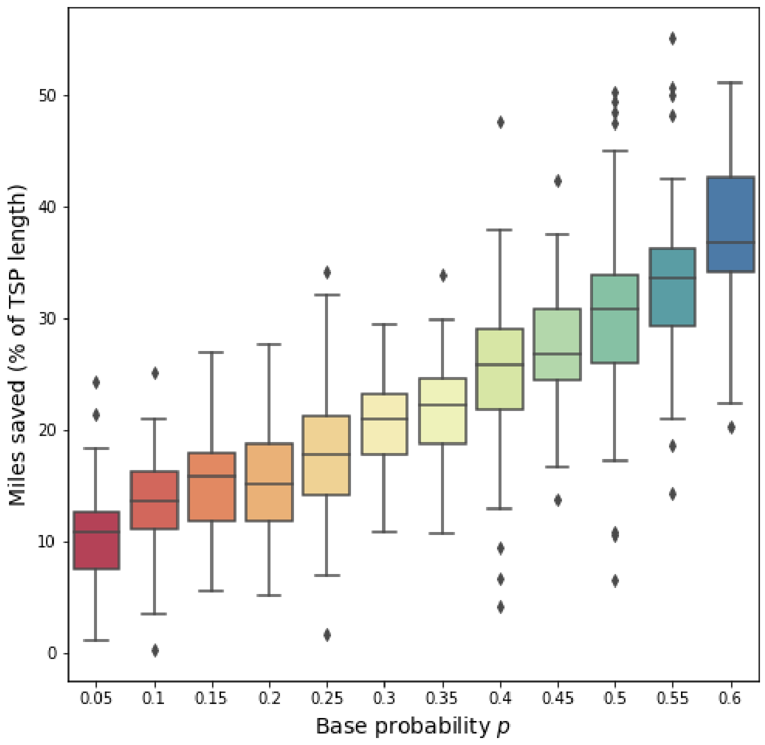

Figure 5: Miles saved as a function of the base probability p used during instance generation. Each box spans the two central quartiles of the distribution over all instances with the given base probability. Whiskers extend to the rest of the distribution, except for outliers which are marked by fliers.

While the above analysis justifies the economical benefits of crowdsourcing end-of-day last-mile deliveries, we also investigate on potential environmental benefits, in terms of miles saved. Let M_{PTSPC} be the length of the tour of the retailer's own vehicle in an optimal solution to the PTSPC and M_{TSP} be the length of the optimal Travelling Salesman tour for the same instance. We define the (percentage) miles saved as $100 \cdot \frac{M_{\text{TSP}} - M_{\text{PTSPC}}}{M_{\text{TSP}}}$. Note that, by definition, the miles saved only depend on the probabilities p_i and not on the amount of fees m_i. Figure 5 shows how the miles saved vary, as a function of the base probability p used during instance generation. Each box spans the two central quartiles of the distribution over all instances with the given base probability. Whiskers extend to the rest of the distribution, except for outliers which are marked by fliers. The figure shows that using crowdsourcing can achieve a reduction in the amount of miles travelled by the retailer's own vehicle which ranges between 10% and 40%. Under the assumption that customers accepting to crowdsource the deliveries need a minimal detour from their planned route, almost all these savings translate into reduced vehicle-miles and into corresponding emissions savings.

5.3 Computational analysis

Table 1 shows an overview of the computational results, comparing different approaches to solve the PTSPC. Each column corresponds to an instance size, from 8 to 20 delivery points (352 instances of each size), while each group of rows refers to an algorithm. We run all experiments on a cluster, in which we reserve one core of an Intel Xeon processor running at 1.7GHz with 4GB RAM.

The first row lists the time needed to enumerate all sets O and, for each of them, compute $\mathbb{E}_A\big[C(O)\big]$. This approach is infeasible in practice because, as expected, time grows exponentially in n; the enumeration takes more than fourteen hours for instances with 20 delivery points. In the next sets of row, we report indicators of the performances of the other algorithms. Rows "Time (s)" denote the elapsed time in seconds. Rows "OptGap%" list the percentage gap between the cost of the best solution found by the algorithm (UB) and the cost of the optimal solution (OPT). We compute the optimality gap as $100 \cdot (\text{UB} - \text{OPT})/\text{UB}$. For the branch-and-bound algorithm we report two more quantities. Row "Gap%" gives the gap computed as $100 \cdot (\text{UB} - \text{LB})/\text{UB}$, where LB denotes a lower bound on the objective value of the optimal solution. This is the gap that the algorithm can report to the user when it does not know the true optimal objective. Rows "Closed%" report the percentage of instances of the given size for which the gap was zero. Row "Nodes" lists the number of nodes visited during the exploration of the tree.

Next, we report the results of the branch-and-bound algorithm, which we run with a time limit of 1 hour. To highlight the importance of upper bound $\bar{z}'$, we present the results both when we do not calculate this bound at the root node (rows "B&B, no $\bar{z}'$") and when we do (rows "B&B"). When not using $\bar{z}'$, the algorithm closes all instances up to size 12; on the other hand, it cannot solve to optimality any instance of size 17 and above. For these latter instances, the algorithm still has large gaps at the end of the runtime (rows "Gap%"). However, because we know (via enumeration) the cost of the optimal solution, we can also give a measure of the quality of the best feasible solution found within the time limit in rows "OptGap%". In this case, we see that the B&B algorithm finds high quality solution, all within 0.10% of the optimal on average. B&B, thus, identifies low cost solutions but has more trouble to prove the optimality (or even the quality) of these solutions because of loose lower bounds. When using bound $\bar{z}'$ computation times increase slightly for the smaller instance sizes. The gaps, however, improve and the algorithm closes all instances up to size 13, while it cannot solve to optimality any instance of size 18 an above. For smaller instances which are solved within the time limit, using $\bar{z}'$ allows to explore fewer nodes before reaching the provably optimal solution. For larger instances, on the other hand, the algorithm explores more nodes before the time limit hits, because the tighter bound allows to enter a node and prune it quickly in more occasions. In general, the gaps with the optimal solution (row "OptGap%") remain small, because they are more influenced by lower rather than upper bounds.

To see the different quality of the bounds directly, Figure 6 shows the upper and lower bound gaps at

		Instance Size n												
		8	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Enum	Time (s)	0.52	5.21	5.07	22.50	49.74	78.94	189.77	308.40	1661.55	6586.03	8888.34	12528.09	50481.00
B&B, no $\bar{\mathbf{z}}'$	Time (s)	1.42	4.57	6.36	13.00	50.04	81.29	238.75	543.99	1506.32	3600.00	3600.00	3600.00	3600.00
	Gap%	0.00	0.00	0.00	0.00	0.00	0.06	1.91	2.79	8.41	18.58	18.78	18.91	19.08
	OptGap%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.05	0.09	0.08	0.10	0.10
	Closed%	100.00	100.00	100.00	100.00	100.00	98.58	90.34	88.14	28.41	1.70	0.00	0.00	0.00
	Nodes	452.55	892.86	1761.02	3508.37	7029.26	13435.47	17198.99	16080.76	22461.92	10590.18	5658.16	5223.32	4677.39
B&B	Time (s)	1.86	5.02	8.59	24.73	56.38	84.91	264.17	602.86	1493.77	3600.00	3600.00	3600.00	3600.00
	Gap%	0.00	0.00	0.00	0.00	0.00	0.00	0.89	1.71	5.05	14.86	16.02	16.79	17.08
	OptGap%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.08	0.08	0.09	0.10
	Closed%	100.00	100.00	100.00	100.00	100.00	100.00	99.72	96.59	84.94	47.16	8.24	0.00	0.00
	Nodes	381.12	633.48	1429.85	2787.03	5600.15	9492.63	14628.17	16674.98	19226.10	15973.72	10083.91	7545.08	7166.25
F-step	Time (s)	0.24	0.91	1.57	1.41	3.57	8.25	16.94	42.33	101.54	235.44	511.41	615.64	1178.77
	OptGap%	0.23	0.21	0.18	0.17	0.12	0.19	0.15	0.16	0.14	0.12	0.20	0.10	0.38
	Closed%	96.88	88.92	88.64	85.51	89.49	84.66	86.36	84.94	77.84	73.58	23.86	22.16	19.03
B-step	Time (s)	0.33	1.27	3.02	7.01	28.83	60.43	87.10	168.81	371.23	891.03	1775.00	2631.10	3587.52
	OptGap%	0.09	0.06	0.06	0.04	0.06	0.05	0.05	0.06	0.05	0.08	0.31	0.41	1.39
	Closed%	94.32	93.75	93.47	93.18	92.33	92.90	93.18	89.49	80.68	74.43	20.74	16.76	2.56
FB-bid	Time (s)	0.28	1.69	1.05	1.84	3.97	7.96	12.86	40.21	155.45	328.82	600.06	983.43	1361.84
	OptGap%	0.23	0.21	0.18	0.17	0.12	0.19	0.15	0.16	0.13	0.12	0.18	0.09	0.38
	Closed%	96.88	88.92	88.64	85.51	89.49	84.66	86.36	84.94	78.13	73.58	24.15	22.44	19.03
BF-bid	Time (s)	0.43	1.27	4.33	9.45	28.21	66.69	107.66	187.82	406.68	956.98	1837.41	2729.43	3592.72
	OptGap%	0.09	0.06	0.06	0.04	0.06	0.05	0.04	0.06	0.05	0.08	0.18	0.40	1.28
	Closed%	94.32	93.75	93.47	93.18	92.33	92.90	93.47	89.49	80.68	74.43	25.85	17.05	3.13
F-MC20	Time (s)	0.09	0.11	0.15	0.13	0.42	0.15	0.24	0.96	0.91	0.83	0.83	1.33	1.46
	OptGap%	0.23	0.21	0.18	0.17	0.12	0.20	0.15	0.18	0.19	0.19	0.22	0.16	0.41
	Closed%	96.88	88.92	88.64	85.51	89.49	84.09	85.23	84.09	58.81	59.38	23.01	18.75	16.19
F-ML	Time (s)	0.26	0.96	1.61	1.71	3.86	9.01	17.88	45.09	109.40	242.15	536.59	607.90	609.18
	OptGap%	0.23	0.21	0.18	0.17	0.12	0.19	0.15	0.16	0.14	0.12	0.20	0.10	0.37
	Closed%	96.88	88.92	88.64	85.51	89.49	84.66	86.36	84.94	77.84	73.58	23.86	22.44	19.60

Table 1: Comparison of different approaches to solve the PTSPC: complete enumeration, the branch-and-bound algorithm presented in Algorithm 1, the four heuristic introduced in Section 4.2, and two versions of heuristic F-step in which we estimate the objective function using the Monte Carlo and Machine Learning (Section 4.3) estimators. Rows "Time (s)" report the runtime in seconds. Row "Gap%" lists the optimality gap computed using lower and upper bounds introduced for the B&B algorithm. Rows "OptGap%" report the gap between the best solution found by the algorithm and the true optimum. Rows "Closed%" list the fraction of instances for which the algorithms identified the optimal solution.

the root node. These gaps are respectively defined as $100 \cdot (\mathrm{UB} - \mathrm{OPT})/\mathrm{UB}$ and $100 \cdot (\mathrm{OPT} - \mathrm{LB})/\mathrm{OPT}$, where UB and LB are the values of upper ($\bar{z}$ and $\bar{z}'$) and lower ($\underline{z}$) bounds computed at the root node of the B&B tree. The leftmost chart aggregates instances by size n, the central one by base probabilities p, and the rightmost one by base fees m. As mentioned above, the figure shows that the lower bound is looser than the upper bounds, and causes large gaps. It also shows that $\bar{z}'$ is significantly tighter than $\bar{z}$. Even if the main issue faced by the B&B algorithm is that the lower bound is weak, a tighter upper bound offers more chances to prune larger parts of the tree earlier and, thus, speed up the tree exploration. This is, indeed, reflected in the results of Table 1.

Figure 6 also shows that the quality of the upper bounds decreases for high probabilities and, to a lesser extent, for high fees. In case of higher probabilities, in fact, more deliveries will be crowdsourced and the term c_V in bound $\bar{x}$ is a bad approximation of the routing costs. When fees are high, optimal solutions tend to exclude the deliveries with the highest fees from the offered set; therefore, the worst-case costs computed by $\bar{z}$ and $\bar{z}'$ can be very far from the optimum. The quality of the lower bound, on the other hand, improves for high probabilities and high fees. As mentioned in Section 2.2, when probabilities of acceptance are 1 the PTSPC reduces to the PTP, which we use to compute $\underline{z}$. It seems reasonable, then, that for higher probabilities the difference between the cost of the solutions of the PTSPC and of the PTP diminishes and the bound becomes tighter. When fees are high, both the solution to the PTSPC and to the PTP used to compute the lower bound will tend to visit many delivery points with the retailer's own vehicle — in general, those with the highest fees. This makes the two solutions look more similar (and, thus, the bounds tighter), compared to when fees are low.

One last remark about the B&B algorithm is that, while the final gaps remain low for all instance sizes, there is a sharp drop in the number of instances solved to optimality starting from size 17 (or even 16 when not using $\bar{z}'$). This drop means that the B&B algorithm still finds very good solutions (small gaps), but not exactly the optimal ones (low "Closed%"). Finally, we note how the number of nodes explored tends to increase up to instances of size 16; from size 17 on, because the exploration of each node starts to become time consuming, the algorithm visits fewer nodes within its time limit.

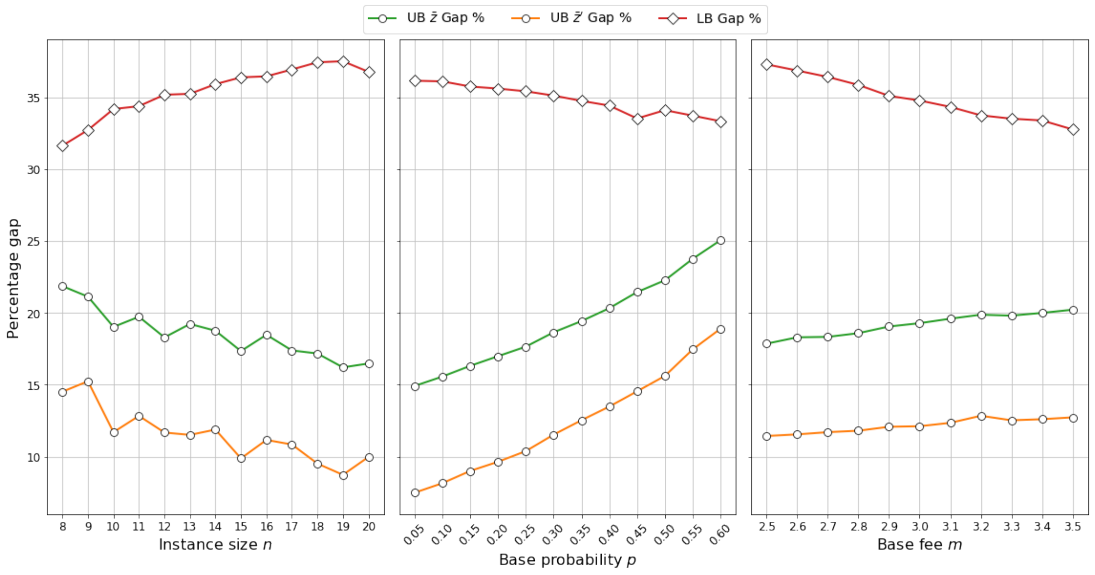

Figure 6: Percentage gaps of the upper and lower bounds, compared to the optimal (or best-known) solution, at the root node of the B&B tree.

Going back to Table 1, the next four groups of rows refer to the heuristic algorithms introduced in Section 4.2. The F-step heuristic is faster than the B-step algorithm, because it tends to visit smaller sets (remember that it starts with $O = \emptyset$). On the other hand, it tends to have larger gaps for all instance sizes but for 19 and 20. Because the optimal solution tends to offer more than half of the deliveries (see Section 5.2) we could, in fact, expect that B-step gives slightly better results. In both cases, however, it is striking how the heuristic algorithms yield low gaps, with the vast majority lower than 1%. We attribute this phenomenon to the shape of the objective function of our problem which, as we observed empirically, tends to flatten once $|O|$ reaches values close to $|O^{\text{opt}}|$. Figure 7 visualises this phenomenon. The figure reports the gap between the objective value of generic sets O and the optimal set O^{opt}, averaged over all sets O of sizes between $|O^{\text{opt}}|-5$ and $|O^{\text{opt}}|+5$ and over all instances. Note that the median gap for size difference 0, i.e., for sets O of the same size as O^{opt}, lies well below 5%. Considering that the heuristics build sets which are locally optimal, in light of Figure 7, it is less surprising that they manage to keep the average gaps below 1%. To summarise, it seems important to determine how many customers should be offered for crowdsourcing and, once this is established, one can get further saving by carefully choosing *which* customers to crowdsource. This is a fact which we can exploit to devise further heuristics, as we discuss in Section 6.

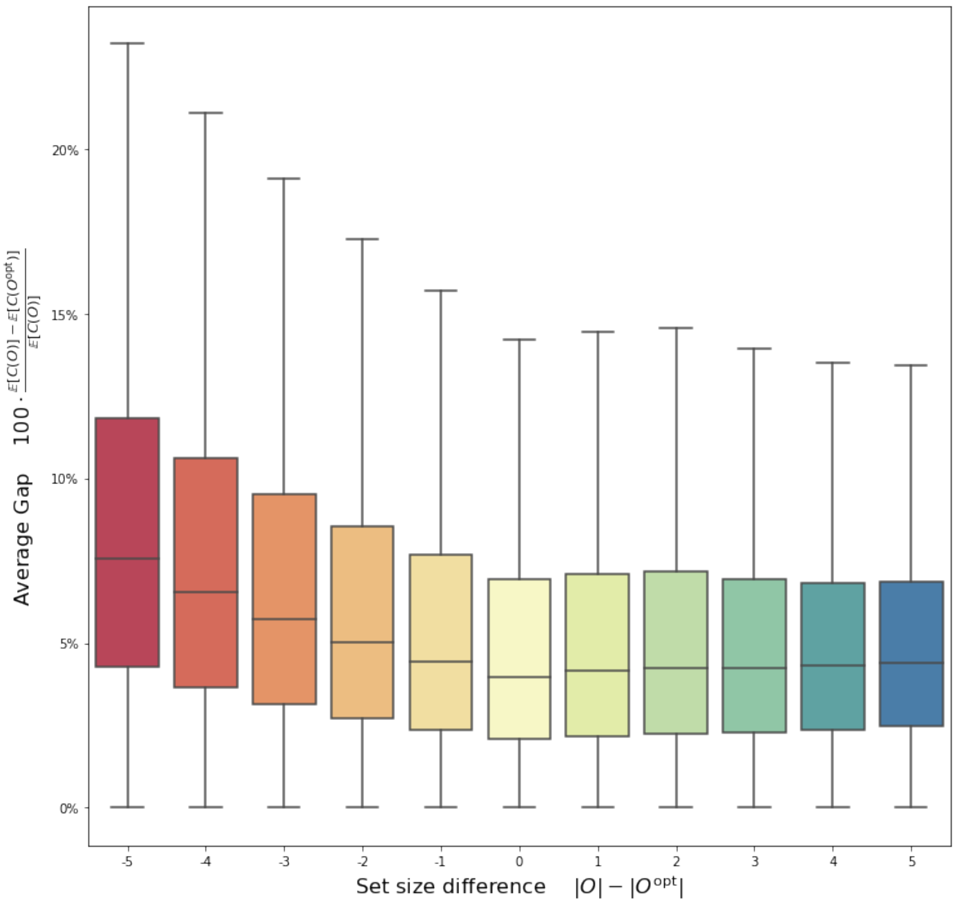

Figure 7: Average gap between the objective value of sets O and the objective value of the optimal set O^{opt}. The figure reports such gap, computed as $100 \cdot \big(\mathbb{E}_A\big[C(O)\big] - \mathbb{E}_A\big[C(O^{\text{opt}})\big]\big)/\mathbb{E}_A\big[C(O)\big]$, for all sets $O \subseteq V$ which have size between -5 and $+5$ compared with the size of O^{opt}. In other words, for all instances, we consider all sets O such that $|O^{\text{opt}}| - 5 \leq |O| \leq |O^{\text{opt}}| + 5$ and compare their cost with the cost of O^{opt}. For each size difference, the boxes span between the first and the third quartile, with the central horizontal line denoting the median value. Whiskers extend to the rest of the distribution (omitting outliers).

Table 1 also shows that there is a sharp drop in the number of instances for which the heuristic algorithms find the optimal solution (rows "Closed%"), once the instance size reaches value 18.

Regarding run times, even the fastest of the two stepwise heuristics (F-step) takes an average of roughly 20 minutes for the largest instances and can be impractical to use in real-life decision support tools. Using the bidirectional heuristics BF-bid and FB-bid increases the run times even further, while providing little improvement in terms of solution quality. Therefore, in the last two groups of rows, we focus on speeding up the solution times: because the F-step heuristic is the fastest of the four and produces high quality solutions, we use it as a base and replace the exact evaluation of $\mathbb{E}_A\big[C(O)\big]$ with its estimation using Monte Carlo (algorithm F-MC20) and Machine Learning (algorithm F-ML) estimators. For consistency with the other results, OptGap% will use the true value of the objective function of the set returned by the algorithms (calculated *a posteriori*), even if the algorithms themselves use its approximation.

For the F-MC20 algorithm, we use a sample size of 20 (i.e., $|\mathcal{A}| = 20$ in the notation of Section 4.3) which allows us to speed up the run time of the algorithm considerably (almost always under two seconds) while maintaining a comparable solution quality. Such speed-up makes this algorithm suitable for interactive decision support tools, in which the decision maker can experiment and perform scenario analysis varying instance parameters such as the fee that the planner is willing to offer.

For the F-ML algorithm we took the following approach. We allocate the first five minutes of run time to building the training set. This means that the algorithm computes the exact expected cost for the smaller sets O during this period. If the algorithm completes within the first five minutes, then it is equivalent to F-step (the corresponding values are in grey in Table 1). On the other hand, if after this time there are still sets O to explore, we train the Machine Learning model using the data collected during the first five minutes and then use the model to estimate the objective value of the larger sets. Because the *Elastic Net* model selected in Section 4.3.2 trains in a matter of fractions of a second, this approach would allow us to have constant run times of roughly five minutes, even for instances larger than the ones considered in this study.

The instance sizes for which F-step took longer than 5 minutes were 19 an 20. In this case, the F-ML algorithm has run times close to 5:10 minutes and gives solution of the same quality as the F-step algorithm. Note that for instances of size 20, the average "OptGap%" is even smaller for F-ML than it is for F-step, indicating that choosing the customer to add using the estimated objective value instead of the true $\mathbb{E}_A\big[C(O)\big]$ gives a better set of offered customers, in the end. Analogously, "Closed%" is smaller for both size 19 and 20. Since the run time of this heuristic tends to be stable (and reasonably small) no matter how large the instance size and the solutions produced are of high quality, it can be used in a non-interactive decision support tool which runs in the background roughly five minutes before the "end of day" period starts at the supermarket.

6 Conclusions and future research

In this paper we have introduced the problem of determining a subset of last-mile deliveries that a company should open for crowdsourcing at the end of the day. We placed this problem in the context of both optimisation of the last segment of the retail supply chain, and in that of TSP problems in which not all customers are visited.

We proposed a branch-and-bound algorithm which has the advantage of being able to provide optimality gaps. Gaps are high for larger instances even when using a time limit of one hour. However, the quality of the solutions found by the algorithm seems to be higher than what the gaps would suggest, due to poor lower bounds which inflate the gaps. Therefore, in future works, we plan to devise tighter lower bounds which can speed up the exploration of the B&B tree, provide stronger optimality guarantees for the primal solutions and allow to solve larger instances. We note that exact algorithms for stochastic routing problems are still limited to solve small instances. For example, the algorithm of Laporte, Louveaux, and Mercure [38] (which is still the state-of-the-art exact algorithm for the PTSP) solves instances with 50 customers, but only 5 of them are stochastic. Even recent B&B algorithms for the PTSP were tested on instances of up to 10 customers [42], 18 customers [4], and 30 customers [3].

We also proposed four heuristic algorithms which only explore a small portion of the solution space. Surprisingly, even simple heuristics such as the forward stepwise F-Step consistently give solutions within 1% of the optimum. We attribute this pleasant property to the shape of the objective cost landscape, which becomes flat once the number of offered deliveries is fixed, for reasons similar to those exposed in Section 4.3 for the concentration of $\mathbb{E}_A\big[C(O)\big]$ around its mean. Using this information, we plan to investigate two-stage heuristics in which we first determine a good size for the offered set and accordingly produce a feasible solution quickly. Then, we look among offered sets of the given size to further lower the solution cost. We also plan to investigate the performance of our heuristics on instances with a radically different topology; for example, instances in which customers are strongly clustered.

Finally, we proposed two methods to approximate the objective function of our problem. Because computing the cost of one solution involves solving an exponential number of TSPs, such approximations give dramatic speed-ups in run times. The Monte Carlo simulation method runs under two seconds, and the Machine Learning method gives an almost constant-time algorithm, whose time limit can be set by the user (we used five minutes in our experiments). The interesting property of such approximations is that they have little impact on the solution quality, compared to the heuristic algorithm to which we applied them. For example, using the F-MC20 algorithm, one could build a real-time decision support tools which enables decision-makers to perform extensive scenario analyses.

An analysis of synthetic instances shows that crowdsourcing end-of-day deliveries has the potential to both achieve savings and to reduce the total miles travelled by vehicles, contributing to a more sustainable last-mile supply chain. Moreover, using customers as occasional drivers can reduce the negative effects of current outsourcing policies, increase trust and promote social engagement. In the future, we also plan to extend our study to applications beyond LMD, e.g., in social care problems where pharmacies ask their customers to deliver medicines to their elderly neighbours.

Acknowledgements

Alberto Santini was partially supported by grant “RTI2018-095197-B-I00” from the Spanish Ministry of Economy and Competitiveness. The other authors are partially funded by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project “POCI-01-0145-FEDER-028611”.

References

[1] Maria Albareda-Sambola, Maarten H Van Der Vlerk, and Elena Fernández. “Exact solutions to a class of stochastic generalized assignment problems”. In: *European journal of operational research* 173.2 (2006), pp. 465–487. DOI: 10.1016/j.ejor.2005.01.035.

[2] Aliaa Alnaggar, Fatma Gzara, and James Bookbinder. “Crowdsourced delivery: A review of platforms and academic literature”. In: *Omega* (2019), pp. 102–139. DOI: 10.1016/j.omega.2019.102139.

[3] Mohamed Abdellahi Amar, Walid Khaznaji, and Monia Bellalouna. “A Parallel Branch and Bound Algorithm for the Probabilistic TSP”. In: *International Conference on Algorithms and Architectures for Parallel Processing.* (Nov. 15–17, 2018). Guangzhou, China: Springer, 2018, pp. 437–448. DOI: 10.1007/978-3-030-05051-1_30.

[4] Mohamed Abdellahi Amar, Walid Khaznaji, and Monia Bellalouna. “An exact resolution for the probabilistic traveling salesman problem under the a priori strategy”. In: *Procedia Computer Science* 108 (2017), pp. 1414–1423. DOI: 10.1016/j.procs.2017.05.068.

[5] Enrico Angelelli, Claudia Archetti, Carlo Filippi, and Michele Vindigni. “The probabilistic orienteering problem”. In: *Computers & Operations Research* 81 (2017), pp. 269–281. DOI: 10.1016/j.cor.2016.12.025.

[6] David Applegate, Robert Bixby, Václav Chvátal, and William Cook. *Concorde TSP Solver.* Version 03.12.19. 2003. URL: https://www.math.uwaterloo.ca/tsp/concorde/.

[7] Claudia Archetti, Martin Savelsbergh, and Maria Grazia Speranza. “The vehicle routing problem with occasional drivers”. In: *European Journal of Operational Research* 254.2 (2016), pp. 472–480. DOI: 10.1016/j.ejor.2016.03.049.

[8] Alp Arslan, Niels Agatz, Leo Kroon, and Rob Zuidwijk. “Crowdsourced Delivery – A Dynamic Pickup and Delivery Problem with Ad Hoc Drivers”. In: *Transportation Science* 53.1 (2019), pp. 222–235. DOI: 10.1287/trsc.2017.0803.

[9] Ronald Ayers and Robert Collinge. *Microeconomics*. Pearson, 2005. ISBN: 9780131489707.

[10] Miguel Barbosa. “A data-driven compensation scheme for last-mile delivery with crowdsourcing”. MA thesis. University of Porto, 2019. URL: https://repositorio-aberto.up.pt/bitstream/10216/124212/2/367287.pdf.

[11] Patrizia Beraldi, Gianpaolo Ghiani, Gilbert Laporte, and Roberto Musmanno. “Efficient neighborhood search for the probabilistic pickup and delivery travelling salesman problem”. In: *Networks* 45.4 (2005), pp. 195–198. DOI: 10.1002/net.20063.

[12] Oded Berman and David Simchi-Levi. “Finding the optimal a priori tour and location of a traveling salesman with nonhomogeneous customers”. In: *Transportation Science* 22.2 (1988), pp. 148–154. DOI: 10.1287/trsc.22.2.148.

[13] Dimitris J Bertsimas. “A vehicle routing problem with stochastic demand”. In: *Operations Research* 40.3 (1992), pp. 574–585. DOI: 10.1287/opre.40.3.574.

[14] Dimitris Bertsimas and Louis Howell. “Further results on the probabilistic traveling salesman problem”. In: *European Journal of Operational Research* 65.1 (1993), pp. 68–95. DOI: 10.1016/0377-2217(93)90145-D.

[15] Neill Bowler, Thomas Fink, and Robin Ball. “Characterization of the probabilistic traveling salesman problem”. In: *Physical Review E* 68.3 (2003). DOI: 10.1103/PhysRevE.68.036703.

[16] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[17] Leo Breiman, Jerome Friedman, Charles Stone, and Richard Olshen. *Classification and regression trees*. Taylor & Francis, 1984, p. 368. DOI: 10.1201/9781315139470.

[18] Vincent Castillo, John Bell, William Rose, and Alexandre Rodrigues. “Crowdsourcing last mile delivery: strategic implications and future research directions”. In: *Journal of Business Logistics* 39.1 (2018), pp. 7–25. DOI: 10.1111/jbl.12173.

[19] Lars Dahle, Henrik Andersson, and Marielle Christiansen. “The Vehicle Routing Problem with Dynamic Occasional Drivers”. In: *Computational Logistics*. Ed. by Tolga Bekta, Stefano Coniglio, Antonio Martinez-Sykora, and Stefan VoSS. 2017, pp. 49–63. DOI: 10.1007/978-3-319-68496-3_4.

[20] Iman Dayarian and Martin Savelsbergh. “Crowdshipping and Same-day Delivery:Employing In-store Customers to Deliver Online Orders”. In: *Optimization Online* (2017). URL: http://www.optimization-online.org/DB_HTML/2017/07/6142.html.

[21] Mauro Dell’Amico, Francesco Maffioli, and Peter Värbrand. “On prize-collecting tours and the asymmetric travelling salesman problem”. In: *International Transactions in Operational Research* 2.3 (1995), pp. 297–308. DOI: 10.1111/j.1475-3995.1995.tb00023.x.

[22] Aashwinikumar Devari, Alexander Nikolaev, and Qing He. “Crowdsourcing the last mile delivery of online orders by exploiting the social networks of retail store customers”. In: *Transportation Research Part E: Logistics and Transportation Review* 105 (2017), pp. 105–122. DOI: 10.1016/j.tre.2017.06.011.

[23] Harris Drucker. “Improving regressors using boosting techniques”. In: *ICML ’97*. Proceedings of the Fourteenth International Conference on Machine Learning. (July 8–12, 1997). Nashville, USA, 1997, pp. 107–115.

[24] Dominique Feillet, Pierre Dejax, and Michel Gendreau. “Traveling salesman problems with profits”. In: *Transportation science* 39.2 (2005), pp. 188–205. DOI: 10.1287/trsc.1030.0079. URL: https://www.jstor.org/stable/25769242.

[25] Matteo Fischetti, Ivana Ljubic, Michele Monaci, and Markus Sinnl. “A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs”. In: *Operations Research* 65.6 (2017), pp. 1615–1637. DOI: 10.1287/opre.2017.1650.

[26] Katarzyna Gdowska, Ana Viana, and João Pedro Pedroso. “Stochastic last-mile delivery with crowdshipping”. In: *Transportation research procedia* 30 (2018), pp. 90–100. DOI: 10.1016/j.trpro.2018.09.011.

[27] Michel Gendreau, Ola Jabali, and Walter Rei. "Stochastic vehicle routing problems". In: *Vehicle Routing: Problems, Methods, and Applications.* Ed. by Paolo Toth and Daniele Vigo. SIAM, 2014, pp. 213–239. DOI: `10.1137/1.9781611973594.ch8`.

[28] Michel Gendreau, Gilbert Laporte, and René Séguin. "Stochastic vehicle routing". In: *European Journal of Operational Research* 88.1 (1996), pp. 3–12. DOI: `10.1016/0377-2217(95)00050-X`.

[29] Chris Groër, Bruce Golden, and Edward Wasil. "The Consistent Vehicle Routing Problem". In: *Manufacturing & Service Operations Management* 11.4 (2009), pp. 630–643. DOI: `10.1287/msom.1080.0243`.

[30] Árni Halldórsson, Gyöngyi Kovács, Julia Edwards, Alan McKinnon, and Sharon Cullinane. "Comparative analysis of the carbon footprints of conventional and online retailing". In: *International Journal of Physical Distribution & Logistics Management* (2010). DOI: `10.1108/09600031011018055`.

[31] Michael Held and Richard Karp. "A dynamic programming approach to sequencing problems". In: *Journal of the Society for Industrial and Applied mathematics* 10.1 (1962), pp. 196–210. DOI: `10.1137/0110015`.

[32] Sin C Ho and Dag Haugland. "Local search heuristics for the probabilistic dial-a-ride problem". In: *OR Spectrum* 33.4 (2011), pp. 961–988. DOI: `10.1007/s00291-009-0175-6`.

[33] Ronald Hocking. "The Analysis and Selection of Variables in Linear Regression". In: *Biometrics* 32.1 (1976), pp. 1–49. DOI: `10.2307/2529336`.

[34] Kuancheng Huang and Muhammad Nashir Ardiansyah. "A decision model for last-mile delivery planning with crowdsourcing integration". In: *Computers & Industrial Engineering* 135 (2019), pp. 898–912. DOI: `10.1016/j.cie.2019.06.059`.

[35] Patrick Jaillet. "A priori solution of a traveling salesman problem in which a random subset of the customers are visited". In: *Operations research* 36.6 (1988), pp. 929–936. DOI: `10.1287/opre.36.6.929`.

[36] Patrick Jaillet. "Probabilistic traveling salesman problems". PhD thesis. Massachusetts Institute of Technology, 1985.

[37] Kafle Kafle, Bo Zou, and Jane Lin. "Design and modeling of a crowdsource-enabled system for urban parcel relay and delivery". In: *Transportation Research Part B: Methodological* 99 (2017), pp. 62–82. DOI: `10.1016/j.trb.2016.12.022`.

[38] Gilbert Laporte, Francois Louveaux, and Hélene Mercure. "A priori optimization of the probabilistic traveling salesman problem". In: *Operations research* 42.3 (1994), pp. 543–549. DOI: `10.1287/opre.42.3.543`.

[39] Gilbert Laporte, Francois Louveaux, and Hélène Mercure. "The vehicle routing problem with stochastic travel times". In: *Transportation science* 26.3 (1992), pp. 161–170. DOI: `10.1287/trsc.26.3.161`.

[40] Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Demetrio Laganà. "The Vehicle Routing Problem with Occasional Drivers and Time Windows". In: *Optimization and Decision Science: Methodologies and Applications.* Ed. by Antonio Sforza and Claudio Sterle. 2017, pp. 577–587. DOI: `10.1007/978-3-319-67308-0_58`.

[41] Giusy Macrina, Luigi Di Puglia Pugliese, Francesca Guerriero, and Gilbert Laporte. "Crowdshipping with time windows and transshipment nodes". In: *Computers & Operations Research* 113 (2020). DOI: `10.1016/j.cor.2019.104806`.

[42] Soumaya Sassi Mahfoudh, Walid Khaznaji, and Monia Bellalouna. "A branch and bound algorithm for the porbabilistic traveling salesman problem". In: *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD).* (June 1–3, 2015). Takamatsu, Japan: IEEE, 2015, pp. 1–6. DOI: `10.1109/SNPD.2015.7176284`.

[43] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis.* Cambridge University Press, 2017.

[44] Michele Monaci, Ciara Pike-Burke, and Alberto Santini. "The 0–1 Time-Bomb Knapsack Problem". In: *Computers & Operations Reesarch* to appear (2021).

[45] Roberto Montemanni, Federico D'ignazio, Xiaochen Chou, and Luca Maria Gambardella. "Machine Learning and Monte Carlo Sampling for the Probabilistic Orienteering Problem". In: *2018*

Joint 10th International Conference on Soft Computing and Intelligent Systems (SCIS) and 19th International Symposium on Advanced Intelligent Systems (ISIS). IEEE. 2018, pp. 14–18. DOI: 10.1109/SCIS-ISIS.2018.00014.

[46] Zongxu Mu, Jérémie Dubois-Lacoste, Holger H Hoos, and Thomas Stützle. "On the empirical scaling of running time for finding optimal solutions to the TSP". In: *Journal of Heuristics* 24.6 (2018), pp. 879–898. DOI: 10.1007/s10732-018-9374-0.

[47] Temel Öncan, Kuban Altnel, and Gilbert Laporte. "A comparative analysis of several asymmetric traveling salesman problem formulations". In: *Computers & Operations Research* 36.3 (2009), pp. 637–654. DOI: 10.1016/j.cor.2007.11.008.

[48] Shenle Pan, Vaggelis Giannikas, Yufei Han, Etta Grover-Silva, and Bin Qiao. "Using customer-related data to enhance e-grocery home delivery". In: *Industrial Management & Data Systems* (2017). DOI: 10.1108/IMDS-10-2016-0432.

[49] Mikko Punakivi and Juha Saranen. "Identifying the success factors in e-grocery home delivery". In: *International Journal of Retail & Distribution Management* (2001). DOI: 10.1108/09590550110387953.

[50] John Quinlan. "Learning with continuous classes". In: *5th Australian joint conference on Artificial Intelligence*. Vol. 92. 1992, pp. 343–348.

[51] Alberto Santini. *alberto-santini/ptspc-instances*. Version 1.0. 2020. DOI: 10.5281/zenodo.4031192. URL: https://github.com/alberto-santini/ptspc-instances.

[52] Hanif Sherali and Warren Adams. *A reformulationlinearization technique for solving discrete and continuous nonconvex problems*. Springer, 1999. DOI: 10.1007/978-1-4757-4388-3.

[53] Hanif Sherali, Subhash Sarin, and Pei-Fang Tsai. "A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints". In: *Discrete Optimization* 3.1 (2006), pp. 20–32. DOI: 10.1016/j.disopt.2005.10.004.

[54] Michael Steele. *Probability theory and combinatorial optimization*. Vol. 69. Siam, 1997. DOI: 10.1137/1.9781611970029.

[55] Roberto Tadei, Guido Perboli, and Francesca Perfetti. "The multi-path traveling salesman problem with stochastic travel costs". In: *EURO Journal on Transportation and Logistics* 6.1 (2017), pp. 3–23. DOI: 10.1007/s13676-014-0056-2.

[56] Ton Volgenant and Roy Jonker. "On some generalizations of the travelling-salesman problem". In: *Journal of the Operational Research Society* 38.11 (1987), pp. 1073–1079. DOI: 10.2307/2582232.

[57] Baris Yildiz and Martin Savelsbergh. "Service and capacity planning in crowd-sourced delivery". In: *Transportation Research Part C: Emerging Technologies* 100 (2019), pp. 177–199. DOI: 10.1016/j.trc.2019.01.021.

[58] Mengying Zhang, Jin Qin, Yugang Yu, and Liang Liang. "Traveling salesman problems with profits and stochastic customers". In: *International Transactions in Operational Research* 25.4 (2018), pp. 1297–1313. DOI: 10.1111/itor.12310.

[59] Mengying Zhang, John Wang, and Hongwei Liu. "The Probabilistic Profitable Tour Problem". In: *International Journal of Enterprise Information Systems (IJEIS)* 13.3 (2017), pp. 51–64. DOI: 10.4018/IJEIS.2017070104.

[60] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the royal statistical society: series B (statistical methodology)* 67.2 (2005), pp. 301–320. DOI: 10.1111/j.1467-9868.2005.00503.x.

A Definition of c_X

We formalise the TSP problem, which we must solve to obtain cost c_X for a given subset of vertices X ($\{0\} \subseteq X \subseteq V$. Let, $E_X \subseteq E$ be the subset of edges with both endpoints in X. Then c_X is defined as the cost of the optimal solution to the following TSP problem:

$$
\begin{aligned}
c_X = \min \quad & \sum_{\{i,j\}\in E_X} c_{ij} w_{ij} && && (27)\\
\text{s.t.} \quad & \sum_{\{i,j\}\in\delta(i)} w_{ij} = 2 && \forall i \in X && (28)\\
& \sum_{\{i,j\}\in\delta(S)} w_{ij} \geq 2 && \forall S \subset X && (29)\\
& w_{ij} \in \{0,1\} && \forall \{i,j\} \in E_X, && (30)
\end{aligned}
$$

where w_{ij} is a binary variable denoting whether edge $\{i,j\} \in E_X$ is part of the tour, $\delta(S)$ denotes the set of edges with one extreme in S and one in $X \setminus S$, and $\delta(i) = \delta(\{i\})$.

B Definition of PTP(X, Y)

We formalise the Profitable Tour Problem (PTP). The PTP is defined on a graph $H = (U', D)$ where U' is the set of vertices and D is the set of edges. Vertex $0 \in U'$ is the depot, while the other vertices form set $U = U' \setminus \{0\}$. Each vertex $i \in U$ has an associated reward $m_i \in \mathbb{R}^+$ and each edge $\{i,j\} \in D$ has a corresponding travel cost $c_{ij} \in \mathbb{R}^+$. A solution to the PTP is a simple tour starting and ending at the depot and possibly visiting vertices in U. The profit of a tour is the difference between the prizes collected at vertices visited by the tour, and the travel costs of the edges comprising the tour. The PTP asks to find the tour with the highest profit.

We are interested in a generalisation of the PTP, in which U is partitioned into two sets, X and $U \setminus X$, and we require the tour to visit all vertices of X. We can solve our version of the problem with an Integer Programme (IP), introducing two sets of variables: $v_i \in \{0,1\}$ which takes value 1 iff the tour visits vertex $i \in U'$, and $w_{ij} \in \{0,1\}$ iff the tour uses edge $\{i,j\} \in D$. A model for the problem is, then:

$$
\begin{aligned}
\max \quad & \sum_{i\in U} m_i v_i - \sum_{\{i,j\}\in D} c_{ij} w_{ij} && && (31)\\
\text{s.t.} \quad & v_i = 1 && \forall i \in \{0\} \cup X && (32)\\
& \sum_{\{i,j\}\in\delta(i)} w_{ij} = 2 v_i && \forall i \in U' && (33)\\
& \sum_{\{i,j\}\in\delta(S)} w_{ij} \geq 2 v_k && \forall S \subset U,\ \forall k \in S && (34)\\
& v_i \in \{0,1\} && \forall i \in U' && (35)\\
& w_{0j} \in \{0,1,2\} && \forall \{0,j\} \in D && (36)\\
& w_{ij} \in \{0,1\} && \forall \{i,j\} \in D\ (i,j \neq 0). && (37)
\end{aligned}
$$

Constraints (32) force the tour to include the depot and all vertices of X, constraints (33) ensure that the tour uses two edges incident to each visited vertex, while constraints (34) are sub-tour elimination constraints. We denote with PTP$(X, U \setminus X)$ the objective value of the optimal solution of (31)–(37).

We point out that one needs not solve (31)–(37) as an IP. In fact, we use the transformation of the PTP into an Asymmetric TSP on an extended graph proposed by Volgenant and Jonker [56]. In the notation used in [24, Sec. 2.2], we can impose the condition that the tour visits delivery locations X by skipping the creation, in the extended graph, of vertices of type v_{n+i} for all $v_i \in X$.