Are some prices more equal than others? Evaluating store-based price differentiation

Hugo Jonker
Open University Netherlands,
Radboud University
hugo.jonker@ou.nl

Stefan Karsch
TH Köln,
stefan.karsch@th-koeln.de

Benjamin Krumnow
TH Köln,
Open University Netherlands
benjamin.krumnow@th-koeln.de

Godfried Meesters
Open University Netherlands

***Abstract*—Online vendors typically offer different stores to sell their items, such as desktop site, mobile site, country-specific sites, etc. Online rumours and news media reports persist that item prices between such views differ. While several academic works have investigated price differentiation, to date, no systematic method for analysing this question was put forth. We devise an approach to investigate such store-based price differentiation, based on three pillars: a framework that can perform cross-store data acquisition synchronously, a method to perform cross-store item matching, and constraints to limit client-side noise factors. We test our method in an initial case study to investigate store effects on flight pricing. We gather pricing data of 824 flights from 15 stores (incl. desktop sites, mobile apps, and mobile sites) over a 38-day period. Our experiment shows that price differences occur frequently. Moreover, even in a limited run we find strong indications of store-specific pricing for certain vendors. We conclude that (i) a larger study into store-based price differentiation is needed to better gauge this effect; (ii) future research in this general domain should take store-based differences into account in their study design.**

I. Introduction

Shopping is a basic fact of life that contains an interesting adversarial relation: the shopper wants to pay as little as possible, while the vendor wants to sell for as much as possible. In brick-and-mortar stores, either side has limited access to information to improve their side of this bargaining process. This changes radically for online shopping: customers can trivially look up prices of competitors, while vendors can leverage assorted technical measures to glean more information about their customers. This allows vendors to tailor their prices on the fly. With respect to this, we distinguish, as is common, between price differentiation and price discrimination. Price differentiation occurs when the same item is priced differently in another situation. Price discrimination occurs when this difference can be attributed to differences in user attributes between those two situations. Thus, all price discrimination is a form of price differentiation, but not vice versa.

Price discrimination has been the subject of various academic studies. In 2012, Mikians et al. [22] found indications of price discrimination using automated scraping. Since then, studies investigated the effect of user attributes (amongst others, device fingerprinting [13], user profiling [12]), explored different methods to collect data (such as Amazon Turk [12], crowd-sourcing [23], [14]), and investigated occurrence of price discrimination in different markets (e.g., airline tickets [27], rental cars [13]).

Less attention from the academic community has been devoted to group-targeted pricing. Several cases have occurred in real life, such as ZIP-code-driven pricing of homework tutoring in the US [1], or price differences between mobile and desktop sites as reported by German travel magazine *Clever Reisen* [17]. Clever Reisen manually checked trip prices once using up to four different stores (mobile and desktop). They found that prices can vary up to 8% between a company's mobile app and their website.

This informal investigation of price differences between two stores of the same vendor poses an especially interesting case: it is highly relevant to online shopping and could be systematically investigated from a user point-of-view, without access to the vendor's internal processes. Moreover, it is quite common for vendors to provide multiple online stores for their items, such as per-country shops (e.g., amazon.de, amazon.fr) or provide multiple versions of the same shop (e.g., desktop/mobile site/app). While such stores cater to specific niches, the vendor could also choose to vary pricing between them. To the best of our knowledge, there has not yet been a systematic investigation into whether price differences occur between different stores of the same underlying vendor. In this paper, we propose a method to systematically investigate store-based price differentiation, provide a proof-of-concept implementation and execute a limited validation test of the proof-of-concept against vendors of flights. For all vendors, we compare app vs. German desktop site, and French site vs. German site. In addition, for one vendor, we compare app vs. German desktop website vs. German mobile site. Even in this limited validation test, we find strong indications of store-specific pricing.

Contributions. We develop a device-independent approach to simultaneous data extraction. The approach relies on using dedicated machines to perform the data extraction, either by accessing the webstore themselves using Puppeteer, or by

Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb) 2023
3 March 2023, San Diego, CA, USA
ISBN 1-891562-87-8
https://dx.doi.org/10.14722/madweb.2023.23011
www.ndss-symposium.org

interacting with a device (e.g., an app on a mobile phone) to do so using a dedicated extractor (for mobile phones, UI Automater and Appium). Data from app/mobile site was collected using smartphones. We test our proof-of-concept implementation by collecting data on a handful of flights from 15 stores of 5 vendors over a period of 38 days. Our analysis of the collected data strongly suggests that store-based pricing occurs for more than half of the investigated vendors.

Ethics. Our case study required to visit multiple stores synchronously by one company. This can have a negative impact. First, booking systems for flights and accommodation tend to make a tentative reservation of items during the booking process to ensure availability of the item offered to the client. This blocks that item's availability for other customers. To avoid this, we only collected data from search pages and never entered further into the booking process. To the best of our knowledge, bare item searching does not affect item availability. Second, our queries could potentially impact prices shown to other subsequent visitors. The low number of queries per day produced by our bots should have negligible influence given the popularity and therefore large user base of these services. Third, we seek to minimise the impact on the services. Each comparison involves simultaneous data acquisition across all involved stores. To avoid hammering the various stores, we only collect data for one comparison at a time. That is, data collection for the various comparisons is triggered sequentially. In addition, we do not issue repeat requests in case of failure to receive response.

Availability. Our Proof-of-Concept implementation as well as our data set are available from GitHub[1] for follow-up research.

II. Related work

A. Online pricing algorithms

Previous studies have empirically investigated online pricing algorithms. Chen et al. [5] studied swelling prices on Uber, which provided insights on the endurance and frequency of such swells, and insights to the used algorithms. To acquire the data for their study, they mimicked HTTP interaction by a mobile app to communicate with the backend, bypassing the need to scrape the app. In another study, Chen et al. [6] attempt to reconstruct pricing algorithms by third-party sellers on Amazon. They fall back to using web scraping for the data acquisition, as the API provided by Amazon enforces restrictive rate limiting. Their findings show the usage of pricing algorithms in some cases and high price volatility. In contrast, Gibbs et al. [10] used data from analytics companies to study pricing algorithms on Airbnb. Their findings show that dynamic pricing is not widely used in this market.

B. Mobile web $\neq$ regular web?

A most pertinent question is whether or not actual devices are needed for data acquisition. Several studies investigated differences between mobile and desktop sites, typically based on faking/emulation. Das et al. [7] investigated access to sensors on mobile devices via JavaScript. They modified OpenWPM [9], a web measurement tool based on the Firefox desktop browser, to resemble a mobile browser. Their findings show that a majority of third-party scripts in the context of advertising or bot detection make use of mobile sensors. A similar approach was taken by Goethem et al. [25], who determined differences in deployment of security measures between desktop sites and their mobile variants. To gain access to mobile sites, Goethem et al. modified a headless Chrome browser to fake the characteristics of a mobile device. Their study showed that there is little difference between both variants in the deployed security measures. They do not report whether they validated the accuracy of their modified Chrome browser compared to sites delivered to real devices.

In contrast to these two studies, Yang and Yue [28] used genuine mobile browsers. They leveraged a modified OpenWPM version to run mobile Firefox on smartphones via the Android debug bridge. Yang and Yue found that measures to disguise a desktop browser as mobile browser can be ineffective in triggering the delivery of a mobile website. Their results show a significant difference in the number of trackers between mobile and desktop sites.

C. Data acquisition

Price studies typically require a dedicated scraper per vendor, limiting their coverage. Several works looked into alternatives. First, to achieve vendor-site agnosticism, Mikians et al. developed a browser extension called $heriff [23]. $herrif facilitates crowd-sourcing price information. Later, Iordanou et al. [14] expanded this initial work to enable price comparisons in a peer-to-peer fashion. Their method identified that 76 out of 1994 services conduct location-based price discrimination. Last, Hannak et al. [12] get real users involved via Amazon's Mechanical Turk. In addition, they investigated mobile browser stores by deploying headless browsers with a manipulated user-agent string. In their experiments, they found two vendors with signs of price discrimination and search steering targeting mobile browser users.

D. Price differentiation

User-related characteristics may affect price differentiation. Mikians et al. [22] found that the user's geo-location, trained personas and the user's origin had an influence on the shown prices in some e-commerce markets. Other factors like the OS or the browser did not lead to any difference in their study. Hupperich et al. [13] investigated what features of a device's fingerprint are most influential for prices. They use a desktop browser and modify the userAgent value to emulate mobile browsers. They found the navigator.userAgent and navigator.vendor browser properties to affect prices the most. Closest to our study is the work by Vissers et al. [27]. They also conducted automated measurements to investigate whether price differences exist on airline websites. However,

[1] https://github.com/godfriedmeesters/diffscraper

their study focused on various user profiles and physical locations, whereas our study concentrates on stores, .i.e., vendor-side differences instead on customer-side differences. Vissers et al. found several price differences, some of which they could attribute to identifiable causes such as tax differences and extreme price volatility for specific items. They also found that search queries more than a minute apart can cause different search results for volatile items. We take this advice into account for our synchronisation approach.

III. Design of a price comparison framework

A price comparison framework is based on two parts: a data collection part, which retrieves item data from the studied vendors, and an item equivalence determination procedure, which determines whether two collected pieces of data concern the same item. An important goal of the data collection part is to eliminate or mitigate any price influences that are not subject of study. The goal of item equivalence is to ensure that only equivalent items are compared, that is, to prevent the proverbial comparing of apples to oranges.

First, which items should be considered equivalent depends on the underlying study. For example, a study comparing prices of round fruits per weight would indeed compare prices of apples to prices of oranges. A study comparing prices of cultivars of apples would distinguish between golden delicious apples, braeburn apples, granny smith apples, etc., whilst considering small and large apples of the same cultivar equivalent. Note that due to EU law,[2] items must be purchasable for the price on display. Thus, additional fees can only concern extras; hiding necessary fees is not allowed. This implies that, for vendors selling to EU citizens, items can be compared once the vendor store displays a price. As item equivalence depends on the specific study under consideration, we detail our interpretation of item equivalence in the discussion of the validation experiment (Sec. IV).

Secondly, the data collector needs to reduce the effect of confounding factors. Here, the fact that this framework studies pricing on different stores comes into play. Various aspects may affect item price beyond the store used to view the item. Care must be taken to mitigate their impact, preferably by eliminating that completely. Alternatively, by ensuring their impact is constant or can be filtered out in another fashion.

A. Data acquisition methodology

To acquire data, the data must be collected from each vendor. This can be done by automated tooling or manually. Both automated tooling and manual data acquisition require upkeep. Manual data acquisition is labour-intense, but this can be overcome by using crowd-sourcing. Previous studies have shown that crowd-sourcing using a browser plugin overcomes limitations of maintaining scrapers and allows to target a broad number of stores [23], [12], [14]. However, this advantage is negated when considering multiple device models (with different screens and resolutions) and classes (mobiles, desktops, tablets). This leads to a wide variety of vendor app/site layouts, which the crowd-sourcing should all encompass. Even more damning is that crowd-sourcing relies on individual's user devices. The vendor could employ price discrimination, which is a confounding factor for studying store-based price differentiation.

Nevertheless, automated scrapers are not perfect, either. As recent studies have shown [15], [16], [18], [4], desktop scrapers tend to be recognisable and recognised scrapers receive different results than regular user browsers. Therefore, measures must be taken to mitigate bot detection. The same applies to mobile scraping using emulated devices [28]. Previous price discrimination studies on mobile devices approached data acquisition by using modified desktop browsers with fitted userAgent strings. However, state-of-the-art bot detection companies use browser fingerprinting techniques [3], [26] and input from client hardware, such as mobile sensors [7], to distinguish bots from real human users. Therefore, to ensure that these aspects do not foul up mobile data acquisition, native devices should be used for the data collection.

Another approach to acquiring data pertaining to mobile phones is to gather pricing data directly from a vendor's backend for mobile users [6]. This can be done in two ways, by scripting user interface (UI) interaction on the client devices, or by faking client interaction on the network level. Scripting human interaction can be done via automation frameworks, such as Selenium, puppeteer or Appium. Much like scrapers for websites, each scraper for a store requires its individual script. In addition, updates to the store's user interface may break scripts and, therefore, the data collection process – again, similar to website scraping. Faking client interaction on the network level has the advantage of skipping user interface aspects. However, network traffic is usually encrypted. While this is not a show-stopper, the process of bypassing encryption becomes cumbersome when the app uses key pinning. We found that even after circumventing encryption, mobile apps typically connect to tens of different API endpoints. Since it is unclear what calls are sufficient to mimic an app, this makes faithful data acquisition via mimicking apps particularly hard. Thus, while mimicking apps can be less error-prone, initial development is far less straightforward than using scrapers, especially when targeting multiple vendors (i.e., unrelated stores/APIs). Moreover, by using UI interaction on client devices, maintenance of data extractors on mobiles remains in the same category as for desktop extractors: monitoring UI changes. Therefore, we make the design choice to script UI interaction for mobile data extraction.

B. Handling confounding factors

As stated, the goal of a framework is to eliminate or mitigate confounding factors. Previous reported aspects can be broadly categorised into four main areas: client-side aspects, timing aspects, contextual constraints, and pricing errors.

Client-side aspects: client-side aspects include effects of browser fingerprinting, client profiling, location, and so on (see

[2]https://europa.eu/youreurope/citizens/consumers/shopping/pricing-payments/index_en.htm

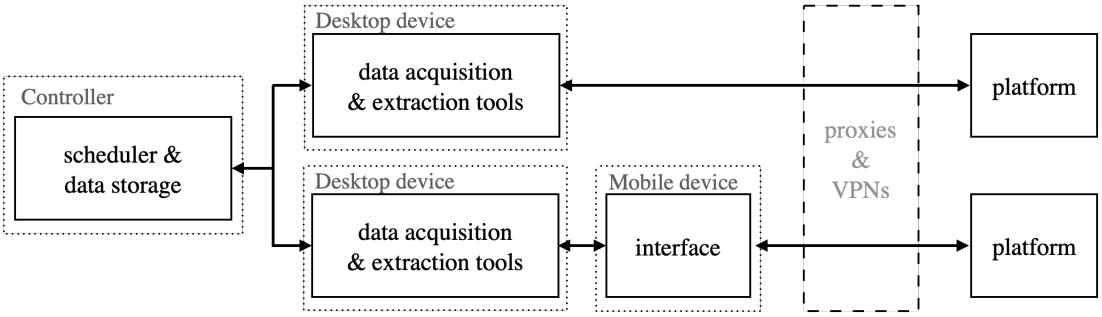

Fig. 1. Framework design

Sec. II). These are necessarily present (even an empty profile is a profile), thus their effects cannot be eliminated. To limit this effect, a price measurement framework should keep client-side aspects constant insofar possible. That is: ensure the same browser fingerprint, start from the same user profile, use the same IP address, etc.

Timing: progression of time affects prices, e.g., expiring items, or tickets for events on specific dates. Vissers et al. [27] established that prices can already differ within a one-minute window. Moreover, increased interest may affect price – one store's data collection may impact prices on another store. Due to network effects, it is impossible to ensure that queries for different stores arrive simultaneously. We point out that the goal of a framework is not to measure the digital world, but to measure the human world. We therefore take the position that best-effort synchronisation is sufficient – if pricing is then still affected, a regular user would also encounter prices affected by random network delays.

Contextual constraints: contextual constraints such as increasing energy prices, tax differences, natural disasters, etc. may affect prices. These should affect stores in the same context equally. For stores in different contexts (e.g., country-specific sites), the difference should be constant across all items. However, context can change over time (e.g., changes in taxes). To account for contextual effects on pricing, a price measurement framework should collect data of multiple items over time.

Pricing errors: pricing errors such as delays in propagating price updates or decimal point errors can cause different prices between stores. Vendors work to catch price errors, thus by gathering item prices over a longer time period, we can eliminate such effects.

C. Framework design and Proof-of-Concept implementation

Based on the problem analysis in above, our system must be able to synchronise heterogeneous scrapers, distributed over multiple devices. It further needs to support data collection on mobile devices and aggregate collected data into a single data repository. This leads to the design depicted in Fig. 1.

The central component is the scheduler that enforces the needed synchronisation. It schedules jobs to orchestrate other components in the system, supplies the information (search query input) to run scrapers (*extraction tools*), and manages a central data repository to persistently store results collected by devices. When a device retrieves tasks to query items from stores, it initiates a scraper. A *scraper* is an automated store-specific data-extracting client. It interacts with one store to request items. In the case of a mobile store, the design supports using an emulator as well as interfacing with an actual mobile device. In the latter setup, the mobile device acts as an interface for the scraper. Session initiation and communication with the store thus happen on the mobile device. Depending on the implementation of the system and target of the study, an external VPN server and/or proxy can be used to control outgoing IP addresses.

To validate the design from the section above, we created a Proof-of-Concept implementation (see [21] for a full, exhaustive description). An overview of our implementation is shown in Fig. 2. The scheduler, *consumers/producers* and scrapers are written in JavaScript and run in a Node.js environment. To facilitate communication between the scheduler and other devices, we use the BullMQ framework[3], a message queue system based on Redis.We use one queue for desktop scrapers and a separate queue for mobile device scrapers. Finally, we use one queue for collecting data from all scrapers. The scheduler stores incoming results in a PostgreSQL database. To control the scheduler and to push new tasks into a queue, we added a command line interface. This interface is used by cron jobs, which automatically push tasks to the workers.

The connection between consumers/producers and their scrapers is facilitated via interfaces in TypeScript. Every scraper is expected to implement this interface, specifically, the methods start, stop, fill search, submit search, store results, and take screenshots. Website scrapers use puppeteer[4] with the stealth plugin[5] for puppeteer. For mobile scrapers, we connected the Android device via the USB port. To control the mobile device's interface from the scraper device, we set up an Appium[6] server on the scraper device and UI Automator on the mobile device. Appium then uses the UI Automator to extract and submit data and to perform UI tasks.

[3]https://github.com/OptimalBits/bull/tree/develop/docs
[4]https://github.com/puppeteer/puppeteer
[5]https://www.npmjs.com/package/puppeteer-extra-plugin-stealth
[6]https://appium.io/

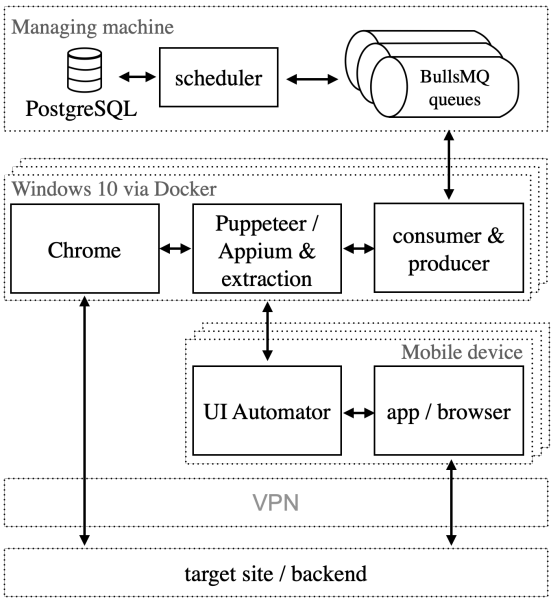

Fig. 2. Proof-of-Concept implementation

D. Client synchronisation

A data collection process involves multiple (at least two) scrapers that execute the following steps:

1) initialise session,
2) fill in searchform,
3) submit search form,
4) retrieve & parse items from result page, and
5) send parsed data to scheduler.

Execution time of each step would vary on a single device between executions already. Our design allows for heterogeneous device classes, which exacerbates this problem. Our PoC implementation addresses this (see Sec. III-B) by adding two synchronisation barriers, before step 2 and before step 3.

As soon as a device is ready for data acquisition, it retrieves a new task from the scheduler and initiates the corresponding scraper (step 1). The synchronisation barrier ensures that the filling in of the search forms is started simultaneously across all scrapers. We synchronise scrapers a second time at the beginning of the data acquisition phase (step 3). That is, we synchronise submission of the search query. Note that due to the centralised design, we approximate simultaneous requests by triggering all scrapers to submit the query simultaneously. Nevertheless, implementation details and run time environments of the individual consumers, scrapers, and target's backend will introduce small timing variations.

IV. Experiment: investigating flight pricing

In order to validate the design, we execute a modest experiment to investigate store-based price differentiation in flight tickets. We choose to focus on airline tickets, as these often are available via multiple stores (app, desktop site, mobile site, reseller). Moreover, while Vissers et al. [27] have not found any evidence of price discrimination in this domain, *Clever Reisen* [17] claims to have witnessed price differentiation here. Besides validating viability of the design, the main goal of our experiment is to determine whether a large-scale investigation is warranted. To that end, we run our experiment for five different popular[7] travel companies that offer flights tickets over a period of 38 days.

A. Data acquisition

To collect data, we select two airlines and three travel agencies. We use all of these companies to search for airline tickets. As input data, we consider cities in Europe and pick random dates between July and August 2021. Further, we use corresponding options within search masks to query for one-way flights only. We left other options in their default setting. Before running our experiment, we manually checked that tickets are offered under equal conditions. For example, whether one store contains additional services in contrast to another store. We verified this by installing the app or visiting the website to review the default setup. Our check did not reveal any differences between stores (see Appx. A).

For each company, we create scrapers to automatically query items via the mobile app, the mobile store, the German desktop website (*.de) and the French desktop website (*.fr). As we compare items retrieved from mobile websites exclusively for one company, our experiment covers 15 stores in total. With these store scrapers we construct comparisons by running at least two different scrapers synchronously. For comparisons that include mobile apps, we use German localisation. To do so, we visit the company's .de domain with our website scrapers. For the mobile scraper, we set the system language to German and download the corresponding localised mobile application and set the phones GPS location to a place in Germany, co-located with the town of the IP-address origin.

For each two-way comparison, we collect data thrice daily between 25 May and 2 July 2021, for a maximum of 114 data points for comparison. Note that to be used in a comparison requires all involved scrapers to be successful, which underlines the fragility of the data acquisition process. Scraper failure, a regular occurrence, happens due to frequent layout changes, changes in flights on offer, as well as network hiccups. This necessitated frequent scraper updates. We also run one three-way comparison (app/web/mobile). Since this comparison requires 3 simultaneous successfully scrapes for one comparison point, we foresee a dearth of comparison points. To mitigate this, we run these scrapers longer, until we collect over 70 comparison points. They ran from 25 May till 5 July (maximum of 123 comparison points). We collect items shown on the result page without following the booking process to avoid blocking items for other users of the store.

Finally, during each run, we took screenshots of all result pages, as well as all recording HTTP responses that contain JSON-formatted data. A random selection of these were then manually compared with the collected data. This verification step found no errors in the automatically collected data.

[7]With the exception of airfrance.de (top 240K) and kayak.de (top 140K), all domains are listed in the Tranco top 100K [20] (ID K2Z6W).

TABLE I
OVERVIEW OF FLIGHT DATA SET

company	scrapers	date		orig–dest	#comps.	#trips	
Air France	app/web	July	01	FRA–CDG	73	2	(2)
		August	01	FRA–CDG	72	2	(2)
		August	09	VIE–AMS	72	3	(3)
	.fr/.de	July	01	FRA–CDG	68	2	(2)
Eurowings	app/web	July	11	AMS–HAM	8	1	(1)
		August	12	CGN–LON	7	3	(0)
	.fr/.de	July	11	AMS–HAM	11	1	(1)
Expedia	app/web	July	01	BRU–AMS	29	24	(3)
		August	10	AMS–ARN	95	7	(6)
		August	18	OPO–BRU	101	20	(6)
	.fr/.de	July	01	BRU–AMS	74	5	(4)
		August	01	BRU–AMS	83	4	(4)
KAYAK	app/web/mob.	August	18	OPO–BRU	71	154	(4)
	app/web	August	07	MAD–FCO	109	247	(13)
		August	13	BER–BCN	101	273	(6)
Opodo	app/web	July	01	FRA–CDG	99	9	(9)
		August	01	FRA–CDG	90	18	(7)
		August	23	CGN–PRG	103	16	(1)
		August	18	OPO–BRU	101	17	(4)
	.fr/.de	July	01	FRA–CDG	101	9	(9)

B. Determining item equivalence

To compare items, we must identify equivalents items from two (or more) stores. We conduct multiple steps to determine whether items are equivalent: attribute normalisation, relevant attributes, and robustness against minor attribute changes. We point out that the choices made here significantly affect what items are considered equivalent, and thereby, the nature of the study. In our case, we aim to determine the price of a trip between two airports. If departure and arrival airport are equivalent, and departure and arrival time are equivalent, we consider the items equivalent. This assumes that no two flights would leave simultaneously from one airport *and* arrive simultaneously at the same destination airport. Obviously, this is not true for code share flights, but these we consider the same flight.

To determine item equivalence for our setting, we first normalise attribute values ('Heathrow', 'London – Heathrow', and 'LHR' $\rightarrow$ `LHR`). More specifically, we check all values of the collected attribute and manually map these to normalised values. For numeric attributes, we establish a standard format and map all values to the standard format. Secondly, we decide on the set of attributes to include for determining item equivalence. We chose the set {date, time, departure airport, arrival airport}. Omitting attributes such as flight number and airline helps to unify code share flights. Lastly, shifts in departure/landing time occur. These constitute different items, which we easily identified and manually matched with their originals throughout the data set. This makes the data set robust against minor changes in these attributes.

C. Resulting data set

Tbl. I shows which vendors/stores were visited, describes the used input search queries, and describes the data points collected. The #trips column shows the total number of unique trips (modulo schedule updates) found. Each trip is one journey from departure to arrival airport, including any intermediate stops. In parenthesis, this column shows the number of equivalent trips found across multiple stores simultaneously. We only compare prices if we extracted data from multiple stores simultaneously at least once. In some cases, there are vast discrepancies between these numbers. We found three major effects for this: first, for vendors who return many trips, much less trips are visible on mobiles (both app and mobile site) than on the desktop without interacting. Second, some vendors offer multi-stage alternatives for direct flights, such as a bus transfer to an alternate departure airport, or a train trip. These two effects explain most of the discrepancy. Lastly, there are some trips that only occur once or a few times. We typically see these only on one store (possibly also due to the first effect). We include the number of trips found on more than one store as only these can be used to analyse store-related price differences. Finally, the #comps column counts runs where all involved scrapers successfully retrieved their result page (column results). These are the data points that can be used for data analysis – the number of data points for each trip for which we collected data simultaneously from multiple stores. Note that these may not be uniformly distributed over the trips found on multiple stores.

V. ANALYSIS

The goal for our analysis is to evaluate whether our approach can find any cases of store-based price differences. In addition, we want to attribute any found differences to likely causes. Our analysis proceeds in three stages. First, we consider, for each vendor, the distributions of relative differences between stores (via boxplots). Second, we consider patterns in pricing via a heatmap. Third, we consider price differences for specific trips, plotting their price developments on both monitored stores.

A. Occurrence of store-based price differentiation (Fig. 3)

We collected sufficient data for analysis for AirFrance, Expedia, Kayak, and Opodo. Fig. 3 shows the distribution of relative price differences found between their (.de) desktop site and their other stores. We found some price differences for each store, though for AirFrance and Expedia these appear minor / incidental. In contrast, the distributions of Opodo and Kayak show clear favouritism.

a) Opodo: French site almost always more expensive: Even our data set provides a limited window, for Opodo, we find that prices on the French website are typically substantially higher than on the German website (median: +14.2%, InterQuartile Range: +10.1%–+22.4%). Indeed, virtually always a lower price was offered on the German website. In Opodo's app, this effect is also present, but smaller (median: +2.25%, IQR: +0.01%–+6.15%). Moreover, while outliers occur in either direction, those favouring the German site over the app are quantitatively larger and occur more often. This is confirmed when looking at a single item, e.g., Fig. 4: users of the French site pay, in this case, tens of euros more for the

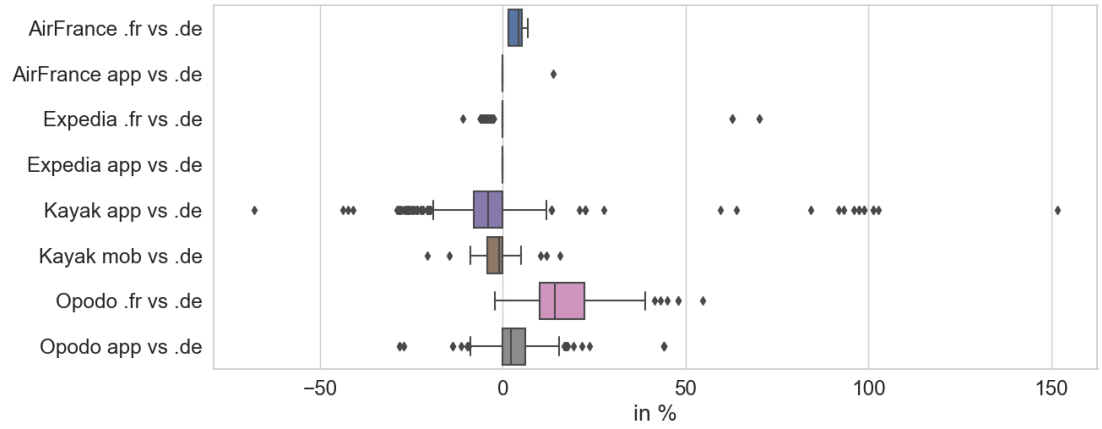

Fig. 3. Relative difference vs .de desktop site

same ticket. On 22^{nd} of June, this was even over €55. While lower prices do occasionally occur on the French site, on the whole, our data set suggests Opodo customers are better off using the German desktop site for booking.

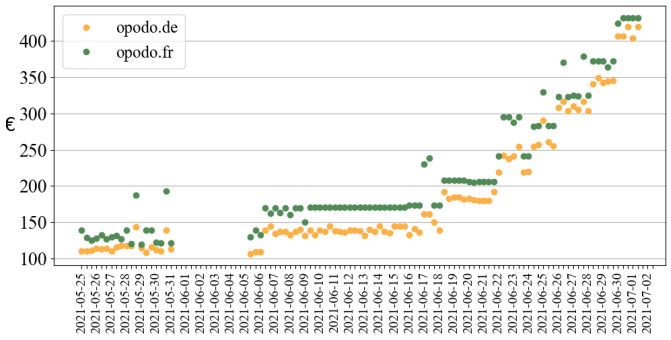

Fig. 4. Opodo, .de vs. .fr, FRA – CDG

b) Kayak: wild variations between stores: For Kayak, we find on average, mobiles get better prices. Prices on all stores vary; in-app prices are mostly slightly better than on the desktop site (median: −4.3%, IQR: −8.0%–0.0%). However, wild outliers occur in either direction. Lucky bookers will be 68% cheaper; unlucky bookers will see prices that more than double (over 150% higher) than on the desktop site. Kayak's mobile site also offers better prices (median: −1.6%, IQR: −4.4%–0.0%). Outliers for the mobile site are less wild, ranging from −20% to +15.7%. On the whole, the best bet for low prices from Kayak is (by a small margin) their app. Its users might even get lucky with a very low price, but should be wary of expensive outliers.

B. Pattern and outlier analysis (Fig. 5)

To evaluate whether there are patterns in the found price differences, we construct a heat map of these (Fig. 5). We omit flights with less than 10% success in data acquisition. Each box shows the relative difference of an item's prices between two stores, with time of data acquisition on the x-axis. Lack of a box means insufficient data was acquired; a lightgray box means that both stores showed the same price. All other boxes denote price differences. For visual clarity, all boxes with data are marked with an edge. Extremes use the 50% colour, so as not to let outliers dominate the colour space.

Using relative percentage differences instead of total prices provides a compact view on the data set. At the same time, differences for trips with higher prices do not overshadow differences of low fare trips. On the downside, relative differences result into higher percentages for increases than for decreases.[8] Hence, this view can provide points of interests, but conclusions require further analysis. Below we discuss high-level observations from this heatmap and indicate which need further inspection.

a) Cross-vendor observations: Any cross-vendor effect manifests as vertical effect in the heatmap. This includes gaps in data acquisition. First, our data acquisition failed to get data from any app from the 5^{th} of June till the 8^{th} of June. This shows up in the heatmap as a vertical gap for all app comparisons. Secondly, other effects besides gaps in data acquisition also occur vertically. An interesting find is that from Fig. 5, we can rule out taxes as a cause for the observed price differences between .de and .fr sites. If taxes caused price differences, this effect would show up in all affected data points, translating to a vertical effect in the heatmap; no such effect is present. This is further strengthened by observing that most Expedia .fr comparisons show no price difference; the exception being when booking date is close to departure date.

b) Vendor-specific observations: With respect to specific vendors, we observe several effects in Fig. 5. First of all, we see that the AirFrance .fr site seems to have an almost-constant price increase over their .de site. This could relate to some type of fee. Secondly, for the Opodo app, we see a wave pattern indicating frequent switches between higher and lower prices. We will consider both cases in more depth in the next section.

c) Perspective on outliers: The heat map also provides more details about recorded outliers. We note that outliers for Expedia (cf., Ex .fr in Fig. 3) are grouped (i.e., on consecutive

[8]For example, a change from €10 to €15 is an increase of 50%, while from €15 to €10 is a decrease of 33%.

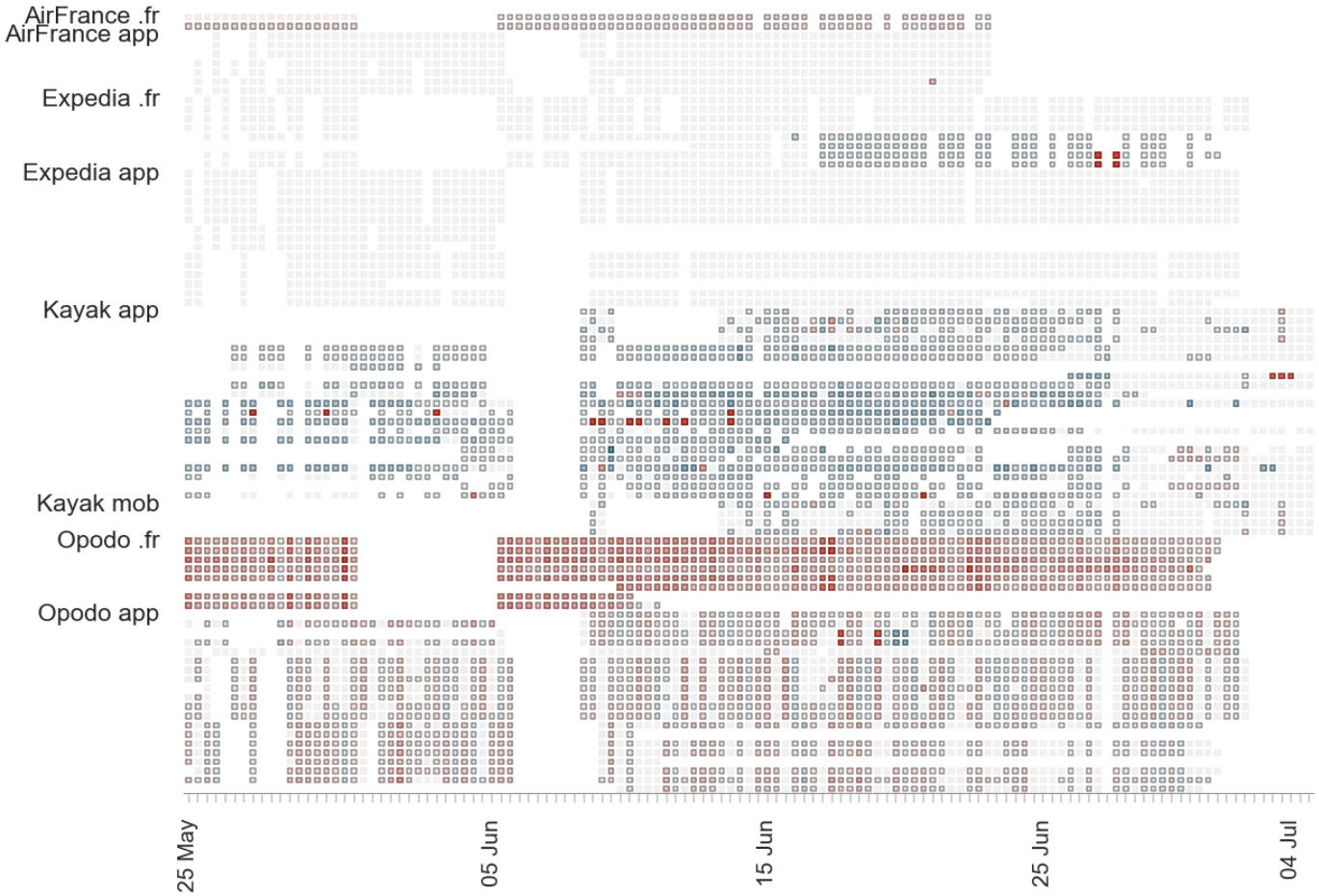

Fig. 5. Overview of relative price differences (colours of extremes truncated to $+/-50\%$)

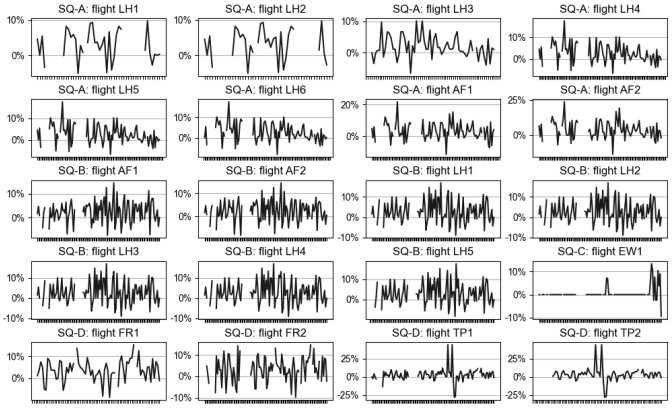

Fig. 6. Opodo website vs. app, differences per item

dates). To find the cause for this pattern, a deeper investigation is necessary (see below). For AirFrance .fr, AirFrance app, and Opodo app, the fraction of outliers is tiny. We ignore such single point outliers, as these could be due to expected errors given the nature of our experiment (see Sec. III-B). Instead, we focus on more consistent data.

C. Analysis of specific cases

a) Opodo's price differences flip frequently (Fig. 6): The perceived wave pattern in Opodo's part of the heat map spans over all four search queries, with the exception of one item. We checked that these price differences are significant (p-value$\leq$0.0001, Wilcoxon signed-rank test).[9] This raises the question whether differences in item prices follow a predictable pattern. Fig. 6 shows, for each item of each search query (SQ-A through SQ-D), the relative difference between website and app price. Breaks in the plot occur where data is lacking. For each query with multiple items, we can find some items whose plots are similar, such as SQ-A: LH1 and LH2, SQ-B: all plots, SQ-D: TP1 and TP2. On the other hand, each of these also has plots which are dissimilar.

Note that none of these patterns re-occur for items from different search queries. We only have four queries, none of which show the same pattern. Data on more queries would be needed to put any specific inference on firm footing. Nevertheless, a common thread already evident is that most items show a high frequency of price swings – hence the wave.

[9]We use Wilcoxon signed-rank test, as our data set is not normally distributed.

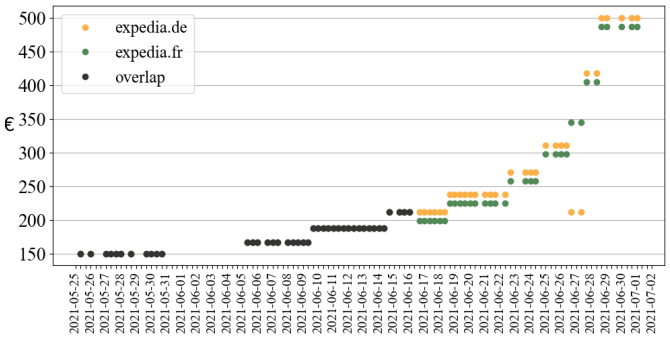

Fig. 7. Expedia, .de vs. .fr, BRU – AMS. Data points under overlap denotes equal prices on both stores

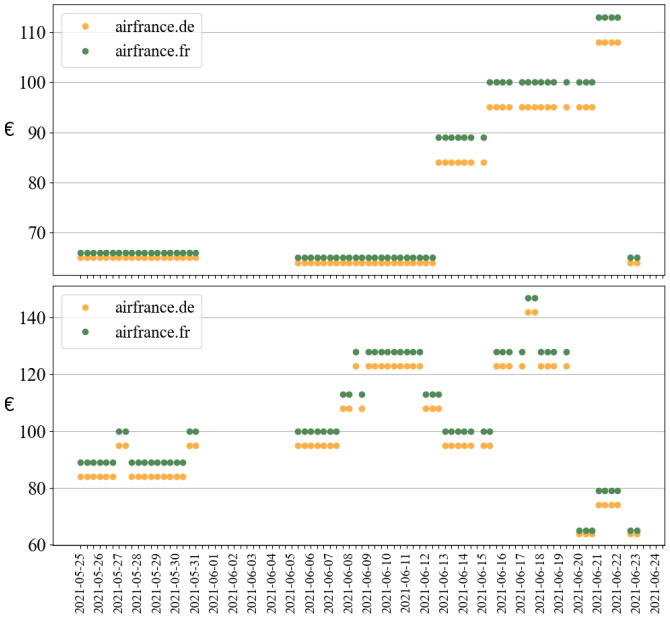

Fig. 8. AirFrance, .de vs. .fr, FRA – CDG, AF1 top; AF2 bottom

This suggests that, in general, checking prices on Opodo's other store can save around 10% of the item price.

b) Expedia: bookinglast minute? Try switching stores (Fig. 7): In the heat map, we see that some items on expedia.fr have lower prices than their .de counterpart in the last third of our observation window. Interestingly, the other four rows within this comparison category show no differences. One example with differences is depicted in more detail in Fig. 7. Here, flight prices start to differ by a constant rate 15 days before departure (except for two outliers). Customers on expedia.fr pay €13 (initially: −6.1%) less than on expedia.de for these 'last-minute' tickets. The four search queries that show no differences in the heat map all have later departure dates (August 1); for these, we lack similar 'last-minute' data. As previously remarked, tax differences between countries would show up in many more points than observed, so these cannot explain the observed differences.

c) €1 or €5 higher prices on airfrance.fr (Fig. 8): For two AirFrance flights, we have sufficient data to compare prices between .de and .fr sites. Both flights showed exactly two levels of price differences: €1 and €5. Interestingly, there is no consistent temporal component to this (see Fig. 8). For both flights, only prices over €65 are €5 more expensive on the .fr sites. Unfortunately, our data set only contains sufficient data to analyse two flights; the data we have is tantalising and merits followup study.

VI. Limitations

The experiment conducted in this paper has several limitations. First, our validation experiment only aims to show the viability of the framework to support price differentiation studies. As such, we only extracted pricing data from a small number of stores of five European vendors of flights. The conclusions of the experiment should not be extrapolated to other vendors, and, given its scale, one should be careful extrapolating our results to other (non-investigated) flights of the same vendors. Nevertheless, the experiment's results show that even a limited experiment with a proof-of-concept tool can already provide indications of (trends in) price differentiation.

Second, our experiment used blank profiles (reset for every scraping job). Note that this is a limitation only of the experiment; the framework does already support using different profiles (or even browsers), simply by instrumenting a scraper with the desired characteristics as one of the desktop devices. With respect to how blank profiles affect data: although no link between browsing history and flight pricing has yet been found, browsing history is known to affect search results [22].

Third, our experimental setup does not fully eliminate all possible confounding factors. For example, store-specific measures such as user tracking (e.g. [8], [19]), bot detection via fingerprinting [15], [26], or behavioural metrics [11] could still impact results. We argue that the impact of any confounding measures not specifically due to detecting the client as a bot would similarly affect genuine clients starting from the same position as our experiment. Bot detection poses another potential source of confounding factors. Stores could employ bot detection to prevent or thwart unwanted resellers. Specifically, they could block bots, show bots different prices, or show bots fake or joke items. The last two measures can thwart bots, and may even allow the store to (significantly) profit from automated resellers [24].

VII. Conclusion

In this work, we set out to explore the feasibility of detecting store-based price differentiation. In contrast to previous price-differentiation studies, we focus on vendor-side differences in pricing (that is, store-based pricing) and conducted a fully automated study. Our experiment is based on a significantly larger amount of data than previous works in this area. Studying store-based price differentiation requires eliminating or mitigating confounding factors as well as gathering the same item from multiple stores. To this end, we designed a framework to enable cross-device automated data collection from different stores. We implemented a proof-of-concept that uses puppeteer for automating desktop browsers and Appium

plus UI automator for automating interaction with mobile devices.

We test our approach in a study of 5 vendors of flights, across 15 stores, gathering data for 38 days. We find evidence of price differentiation between stores for about half of the data set. For some vendors, this seems incidental. For others, the collected data is suggestive of deliberate price differentiation – sufficiently strong to warrant a more detailed in-depth study. Lastly, for some vendors, price differences are fairly blatant. We found recurring price differences of about €50 between .de and .fr stores of Opodo. This is in stark contrast to data from all other investigated .de and .fr sites, ruling out contextual factors such as tax differences. We also compare between website, mobile app, and mobile site. We find that Opodo's app slightly disadvantages users compared to their desktop site. Kayak's stores can differ substantially in price; we found that on average, their mobile stores (app/mobile site) offer lower prices.

Future work. Our experiments show that a larger-scale study into store-based flight pricing is warranted. This can be done in various dimensions. One such way is to perform a deep-dive into one vendor, gathering data on a significant fraction of their flights. This could e.g. be done for Opodo. Another large-scale investigation is to collect data from many more vendors, to determine if there is an industry-wide trend with respect to store-based pricing. Do note that these studies have different challenges: a vendor-specific investigation must take extra measures to reduce the impact of its queries on pricing, while a multi-vendor study faces the challenge of maintaining a plethora of vendor-specific data extractors. Another interesting direction related to cross-vendor studies is to consider seller-incentives. For example, in some cases, resellers stimulate offering lower prices to mobile devices than those offered to desktop clients [2]. A future study could investigate to what extent such features are used across different vendors.

In our experiment, we studied flight prices. Store-based price differentiation can of course also occur for other types of items. Travel-industry related rental items such as hotel bookings or car rentals make interesting candidates. They also allow for fairly easy item comparison. Moreover, these items also have limited availability and are time-sensitive, which is likely to induce vendors to update their pricing. From there, it is only a small step for vendors to consider tweaking prices for specific stores. Hannak et al. [12] reported finding lower prices for hotel bookings on Apple devices than on Android devices for one out of four vendors. Our framework facilitates performing such studies at a larger scale, for more vendors and across more stores.

In all of such price differentiation studies, data extraction remains a contentious point due to server-side changes. On a conceptual level, a followup study is to investigate how to make data extraction significantly more robust. Lastly, taking a step back, there have been various studies into price differentiation focusing on different aspects. Not all studies insulated their data collection from aspects found to be influencing pricing in other studies. What is sorely needed is a taxonomy of various potential influencing factors and a strategy for isolating them.

References

[1] Julia Angwin, Surya Mattu, and Jeff Larson. The tiger mom tax: Asians are nearly twice as likely to get a higher price from princeton review. https://www.propublica.org/article/asians-nearly-twice-as-likely-to-get-higher-price-from-princeton-review, September 2015. Last access: February 6, 2023.

[2] Booking.com. Mobile rate. https://partner.booking.com/en-gb/solutions/mobile-rate, 2023. Last access: February 6, 2023.

[3] Elie Bursztein, Artem Malyshev, Tadek Pietraszek, and Kurt Thomas. Picasso: Lightweight device class fingerprinting for web clients. In *SPSM@CCS*, pages 93–102. ACM, 2016.

[4] Darion Cassel, Su-Chin Lin, Alessio Buraggina, William Wang, Andrew Zhang, Lujo Bauer, Hsu-Chun Hsiao, Limin Jia, and Timothy Libert. Omnicrawl: Comprehensive measurement of web tracking with real desktop and mobile browsers. *Proc. 22nd Privacy Enhancing Technologies Symposium (PETS'22)*, 2022(1):227–252, 2022.

[5] Le Chen, Alan Mislove, and Christo Wilson. Peeking beneath the hood of uber. In *Proceedings of the 2015 ACM Internet Measurement Conference, (IMC'15)*, pages 495–508. ACM, 2015.

[6] Le Chen, Alan Mislove, and Christo Wilson. An empirical analysis of algorithmic pricing on amazon marketplace. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 1339–1349. ACM, 2016.

[7] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In *CCS*, pages 1515–1532. ACM, 2018.

[8] Peter Eckersley. How unique is your web browser? In *Proc. 10th Privacy Enhancing Technologies Symposium (PETS'10)*, volume 6205 of *LNCS*, pages 1–18. Springer, 2010.

[9] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1388–1401, 2016.

[10] Chris Gibbs, Daniel Guttentag, Ulrike Gretzel, Lan Yao, and Jym Morton. Use of dynamic pricing strategies by airbnb hosts. *International Journal of Contemporary Hospitality Management*, 2018.

[11] Daniel Goßen, Hugo Jonker, Stefan Karsch, Benjamin Krumnow, and David Roefs. HLISA: towards a more reliable measurement tool. In *Proc. 21st ACM Internet Measurement Conference (IMC'21)*, pages 380–389. ACM, 2021.

[12] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings of the 2014 Internet Measurement Conference (IMC'14), Vancouver, BC, Canada, November 5-7, 2014*, pages 305–318. ACM, 2014.

[13] Thomas Hupperich, Dennis Tatang, Nicolai Wilkop, and Thorsten Holz. An empirical study on online price differentiation. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, CODASPY, Tempe, AZ, USA*, pages 76–83. ACM, 2018.

[14] Costas Iordanou, Claudio Soriente, Michael Sirivianos, and Nikolaos Laoutaris. Who is fiddling with prices?: Building and deploying a watchdog service for e-commerce. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, pages 376–389, 2017.

[15] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. Fingerprint surface-based detection of web bot detectors. In *Proceedings of 24th European Symposium on Research in Computer Security (ESORICS'19)*, LNCS, pages 586–605. Springer, 2019.

[16] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Ben Livshits, and Alexandros Kapravelos. Towards realistic and reproducible web crawl measurements. In *Proc. The Web Conference 2021 (WWW'21)*. ACM, 2021.

[17] Lutz Kaulfuß. Flugbuchung per smartphone achtung! neue abzocke! Clever Reisen! (1/17), January 2017.

[18] Benjamin Krumnow, Hugo Jonker, and Stefan Karsch. How gullible are web measurement tools? A case study analysing and strengthening OpenWPM's reliability. In *Proc. 18th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '22)*, page 16, New York, NY, USA, 2022. ACM.

[19] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. Browser fingerprinting: A survey. *ACM Transactions on the Web (TWEB)*, 14(2):1–33, 2020.
[20] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. 26th Network and Distributed System Security Symposium (NDSS'19)*, pages 1–15. The Internet Society, 2019.
[21] Godfried Meesters. Synchronising distributed scraping. Master's thesis, Open University of the Netherlands, August 2021. http://www.open.ou.nl/hjo/supervision/2021-godfried-meesters-msc-thesis.pdf.
[22] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *In Proc. 11th ACM Workshop on Hot Topics in Networks (HotNets'12)*, pages 79–84. ACM, 2012.
[23] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Crowd-assisted search for price discrimination in e-commerce: first results. In *Conference on emerging Networking Experiments and Technologies, CoNEXT '13, Santa Barbara, CA, USA, December 9-12, 2013*, pages 1–6. ACM, 2013.
[24] Stefan Schwinghammer. Bonkers vs bots: How to get rid of re-sellers. https://www.soloskatemag.com/bonkers-vs-bots, 2019. Last access: February 6, 2023.
[25] Tom van Goethem, Victor Le Pochat, and Wouter Joosen. Mobile friendly or attacker friendly?: A large-scale security evaluation of mobile-first websites. In *AsiaCCS*, pages 206–213. ACM, 2019.
[26] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers. In *Proc. 2nd NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWEB'20)*, pages 2–14, 2020.
[27] Thomas Vissers, Nick Nikiforakis, Nataliia Bielova, and Wouter Joosen. Crying Wolf? On the Price Discrimination of Online Airline Tickets. In *7th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs'14)*, 2014.
[28] Zhiju Yang and Chuan Yue. A comparative measurement study of web tracking on mobile and desktop environments. *Proc. Priv. Enhancing Technol.*, 2020(2):24–44, 2020.

APPENDIX

A. Settings in search masks

We compared the default settings in each store's search mask. The results are summarised in Tbl. II. There are additional services that can be used within the search mask when using AirFrance. This option is disabled by default on desktops, and it is unavailable in the app. Eurowings is the only vendor that does not offer fare selection via its search mask. Instead, we ensured to use prices shown on the result pages, which always used the lowest fare in our cases. For Opodo, we found that users can select between different restrictions on stops on their trips. However, no restriction is the default option for all checked store.

TABLE II
OVERVIEW ON DEFAULT SETTINGS IN EVALUATED SEARCH MASKS

Airline	Store	Route	Travellers	Fare	Custom options
AirFrance	.de	round-trip	1 adult	economy	bluebiz off
	.fr	round-trip	1 adult	economy	bluebiz off
	app	round-trip	1 adult	economy	n/a
Eurowings	.de	one-way	1 adult	n/a	n/a
	.fr	one-way	1 adult	n/a	n/a
	app	one-way	1 adult	n/a	n/a
Expedia	.de	round-trip	1 adult	economy	n/a
	.fr	round-trip	1 adult	economy	n/a
	app	round-trip	1 adult	economy	n/a
Kayak	.de	round-trip	1 adult	economy	0 bags
	mob.	round-trip	1 adult	economy	0 bags
	app	round-trip	1 adult	economy	0 bags
Opodo	.de	round-trip	1 adult	economy	No stop restrictions
	.fr	round-trip	1 adult	economy	No stop restrictions
	app	round-trip	1 adult	economy	No stop restrictions