UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

A GLOBAL OPTIMIZATION MODEL FOR THE SHOPPER MATCHING PROBLEM IN A DISTRIBUTED LOGISTICS APPLICATION

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN GESTIÓN DE OPERACIONES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

SEBASTIAN ANDRES GUERRATY KORNER

PROFESOR GUÍA:

FERNANDO ORDOÑEZ PIZARRO

PROFESOR CO-GUÍA:

ANDREAS WIESE

MIEMBROS DE LA COMISIÓN:

Richard Weber Haas

Ian Bortnic Kreisberger

Este trabajo ha sido parcialmente financiado por Cornershop

Santiago de Chile

2022

RESUMEN

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE:
Magister en Gestión de Operaciones
RESUMEN DE LA MEMORIA PARA OPTAR AL TITULO DE:
Ingeniero Civil Industrial
Por: Sebastián Guerraty
Fecha: 2022
Profesores guía: Fernando Ordoñez y Andreas Wiese

UN MODELO DE OPTIMIZACIÓN MATEMÁTICA PARA EL PROBLEMA DE ASIGNACIÓN DE SHOPPERS EN UNA APLICACIÓN DE LOGÍSTICA DISTRIBUIDA

Esta tesis presenta una propuesta basada en modelos de optimización matemática para resolver el problema de asignación tripartita de contratistas en la operación de una empresa de logística distribuida.

Cornershop, una solución de mercado de dos lados basada en el uso de aplicaciones móviles requiere asignar de manera recurrente a contratistas para satisfacer las necesidades de los pedidos solicitados por clientes de la aplicación. La empresa utiliza una heurística *greedy* basada en un conjunto de reglas para determinar las asignaciones.

El modelo propuesto se ejecuta en un ambiente de desarrollo en conjunto con una versión local del modelo utilizado por la empresa. Estos dos modelos, junto con dos variantes del modelo propuesto, se corren utilizando escenarios simulados que se basan en los registros obtenidos por la empresa durante la operación.

El modelo propuesto decide que contratista es asignado a cada pedido y en que tienda debe realizarse la compra de ese pedido. El modelo toma en cuenta todas las restricciones que dependen del valor de las variables de decisión, tal como el límite del aforo de contratistas que se pueden asignar a una tienda.

El modelo minimiza los tiempos de viaje de las asignaciones sumado a un costo de un sesgo que refleja la preferencia para la empresa de esa asignación. Las restricciones que no dependen del valor de variables de decisión son aplicadas en preprocesamiento.

El modelo propuesto logra entre una baja de 1% a una mejora de 22% en el costo de asignación en los escenarios utilizados. Adicional a la mejora en eficiencia, el modelo propuesto generalmente logra una mejora de la cobertura de asignación, donde el tiempo promedio en que tarda asignar un pedido disminuye.

Adicional a la evaluación del modelo propuesto, se muestra el resultado del problema relajando restricciones geoespaciales artificiales utilizadas por la empresa para la operación en problemas más pequeños. El modelo con la restricción geoespacial relajada logra una mejora contra la versión con la subdivisión.

Finalmente, se presenta una discusión sobre las diferencias del modelo propuesto, incluyendo algunas de las potenciales limitaciones de su uso. También se presentan propuestas para trabajo futuro que expanden sobre una futura implementación del modelo propuesto.

ABSTRACT

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE:
Magister en Gestión de Operaciones
RESUMEN DE LA MEMORIA PARA OPTAR AL TITULO DE:
Ingeniero Civil Industrial
Por: Sebastián Guerraty
Fecha: 2022
Profesores guía: Fernando Ordoñez y Andreas Wiese

A GLOBAL OPTIMIZATION MODEL FOR THE SHOPPER MATCHING PROBLEM IN A DISTRIBUTED LOGISTICS APPLICATION

In this thesis a mathematical optimization model is proposed to solve a tri-partite assignment problem for a distributed logistics company.

Cornershop, a two-sided market app-based platform, must allocate contractors to perform the jobs needed to fulfil orders placed by customers. The company uses a rule based greedy algorithm to assign the contractors.

The proposed mathematical optimization formulation is run in a local environment against a local version of the heuristic used by the company, for simulated scenarios based on from the company's operation.

The proposed formulation decides the assignment of contractors to a store for a given order, it considers constraints defined by the company that depend on the values for decision variables, such as controlling for store contractor capacity.

The proposed model minimizes travel times plus a skew cost that reflects the preference for that allocation. Constraints that don't depend on the value of decision variables are filtered in pre-processing.

Along with the proposed model, two variants of the mathematical optimization model are benchmarked alongside the company heuristic.

The proposed models achieve from a loss of 1% to a gain of slightly more than 22% of the allocation cost compared to the company heuristic in the tested scenarios. The proposed model also achieves better allocation coverage, where less jobs are left un-allocated for less times compared to the heuristic.

Comparing the approaches for solving the contractor assignment problem, a larger scope problem is benchmarked for the proposed model. The larger scope ignores geospatial subdivisions used by the company. The model without the geospatial partitions achieves an efficiency improvement over the already enhanced result of the proposed model.

Finally, a discussion on differences between the assignment models is presented, discussing some of the limitations of the mathematical optimization approach, alongside with proposals of avenues for future work that build upon a suggested implementation of a version of the proposed mathematical optimization-based models.

ACKNOWLEDGMENTS

I would like to thank my family for their support throughout my education, particularly my parents who have been keen on providing me with the best education that I have had access to.

I would like to thank my friends who were there for support no matter how tough the semester got, they were always there to remind me that we were in it together, it would have certainly been a lot harder path to get to this point without your support. I would like to particularly thank Javi for her unwavering support on whatever I seem to set my mind on.

I would also like to thank the teachers that helped me in achieving not only academically but also helping me find a way to help myself. I'd like to particularly thank Alvaro and Feña for their help on not only becoming a better engineer but also a better person however tough the feedback might have seemed.

I was certainly very lucky to find teachers that would be willing to help you exploit the things I was interested in, many times regardless of how uninterested I might have seemed on the subjects they taught at the time.

Last but not least, I'd like to thank my supervising professors for their guidance during these past couple of years and the dispatcher team at cornershop for trusting me and having the patience so that I could work on this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Platform based services have recently been a growing phenomenon in logistics. Some of the most well-known of these services address the last mile component of the delivery chain, which has traditionally been the least cost-effective component of delivering goods to end customers. Amongst the platform-based providers of last mile deliveries there are those that use a pool of *gig economy* contractors to fulfil their obligations to customers.

Companies that rely on *gig economy* resources to delivery their offering usually act as a platform of a two or more-sided market which essentially coordinate available resources instead of providing the services directly. The approach of a platform solution often achieves less cost and provides more flexibility that what a traditional operation would allow.

Perhaps the most well know platform business models, are those of urban transportation with Uber and Didi being the most recognizable ride hailing platforms. The fulfillment of goods deliveries by providing last mile services has also seen very rapid growth and proved its value by allowing unprecedented levels of scalability during the start of the SARS-COV2 pandemic.

Using a crowd sourced delivery platform, these services have allowed both corporate and "mom and pop shops" unprecedented access to delivering goods to customers with a fast and reliable delivery method, without having to build and scale an in-house solution.

Of the companies providing a distributed last mile service in the Americas, there are those that act as a capability expansion for traditional delivery services, such as Amazon's Flex, where partners only participate in a portion of the delivery workflow, and those that act as a platform for a complete fulfilment cycle, where a partner will approach a store, pick items, trouble shoot issues and deliver.

In the platform delivery category, Instacart, Door dash and Uber Eats are the main participants in the North American market with prepared meals and grocery delivery respectively. Cornershop, a Chilean based company has also positioned itself as a relevant competitor in the goods and groceries delivery business, particularly in Latin America.

At the core of the competitive advantage of these companies in contrast to traditional logistic model counterparts, is the efficient use of the aforementioned distributed resources. This thesis suggests a mathematical optimization model for the optimal allocation of contractors for Cornershop.

1.1 ORDER AND CONTRACTOR MATCHING

Cornershop provides a two-sided market platform for people to order goods (primarily groceries) through digital channels. The company then connects with a contractor workforce to offer the jobs necessary to fulfil these requests for goods placed through the platform.

In a similar fashion to how meal delivery services operate, the company offers customers a pricing structure that is not directly tied to that of the compensation for associates (referred to as *shoppers*

for this thesis). This presents an opportunity in that any efficiency gains achieved through the allocation of jobs is a direct benefit for the company.

The algorithm used for matching the activities related to each side of the platform is a key enabler for the company to operate efficiently. Logistic costs as well as customer and *shopper* experience attributes are also heavily affected by the quality of the match offered to the contractor workforce.

Cornershop uses a greedy algorithm that individually attempts to find the "best" *shopper* match for each order batch. This thesis proposes an approach based on mathematical optimization that deals with many of the potential inefficiencies that happen when using a greedy heuristic such as Cornershop's.

The switch to the proposed approach can potentially have benefits on both operational as well as IT aspects. This thesis focuses on the expected operational impact of switching, though implementation considerations are also briefly discussed.

1.2 CASE STUDY

Given the sensitive nature of a *shopper* selection application for a company that provides a last mile delivery platform, the impact of a mathematical optimization approach discussed in this thesis is conducted within a controlled offline environment. Further validation of results for online or production runs are not considered within the scope.

1.2.1 OBJECTIVE

The main goal is to determine, within reasonable assumptions, the expected change on operational performance metrics for the operation of Cornershop. These metrics can be best described amongst the following categories:

- Efficiency (cost of operating)
- Throughput (how many orders can be fulfilled for a given shopper count)
- Selection fairness (how is shopper effort rewarded)

Although not the main objective of the case study, algorithm performance considerations should also be taken into account. Any proposed solution should mention how it is expected to compare with current solutions from a computation performance perspective.

1.2.2 SCOPE

1.2.2.1 Business practices

The objective of the thesis is to account for problem solution methods, and not to propose new business definitions towards how the company decides to prioritize or allow shoppers to fulfill orders. Any change or recommendation for how the company defines allocation preference or feasibility is only considered as complementary content.

Regardless of there being a more efficient representation of the problem for the company, this thesis will model the problem in as close a way as is possible to how the company has defined the problem. If any enhancements on the representation of the problem are introduced after the fact by Cornershop, these modifications are out of the scope of this thesis.

1.2.2.2 Problem Data

The approach for shopper allocation relies on information provided to the allocation solution which may have inaccuracies or changes over time. The information used for allocation is taken at face value and any changes regarding these inputs are not considered within the scope of the case study.

Information used for matching is provided by both external sources such as travel time APIs and tools built by internal teams, such as order content availability or past contractor performance. The accuracy of the information provided is not a part of the scope of this thesis.

1.2.2.3 Implementation and roll out

The case study aims to predict the expected outcome if the proposed model were to be used, any consideration or changes regarding the implementation of the formulations discussed are not within the scope of the thesis. However, there may be references, comments or recommendations regarding future use that involve implementation considerations, but only serve as complementary content.

1.2.2.4 New business avenues

Cornershop is characterized for a fast-paced growth alongside innovation on methods. Changes on the delivery methods alternatives or new markets are not considered a part of this thesis. Any content regarding delivery alternatives is considered complementary content.

1.3 STRUCTURE

This thesis is structured in 5 sections: A review of relevant literature and a brief discussion as to why the type of problem proposed was selected, a description of the problem the company faces and its current formulation with the proposed approach, a description of the methodology and tools used for evaluating the performance of the proposed approach, results of the proposed approach and its analysis, followed by a conclusion as well as a brief discussion on future directions.

The first section has a brief description of the company as well as a description of the context for the problem the proposed model solves. This section also situates the proposed formulation with its relevant literature, covering the assignment problem and its similarities with implementations for hail a ride and meal delivery platforms. This section also discusses an alternative type of problem in that of the deferred acceptance family of problems.

The second section contains a detailed description of the heuristic used by the company to solve the *shopper* allocation problem, as well as the base for the proposed mathematical optimization model. Within the description of the proposed formulation there is also a description as to why the problem was modeled with the proposed model.

The third section describes how the proposed formulation is measured against the company heuristic: this section has a more detailed description of how what the simulations use for input, how they are structured, and the output obtained from the runs.

The fourth section contains the results obtained from the runs described in the previous section, as well as the analysis of said results. This section contains results for the company heuristics as

well as the proposed model along with three variants of the proposed model. This section also contains the results of solving the problem for a larger scope of city-wide models.

Finally, the fifth section of this thesis contains the summary of key findings as well as suggestions for future work that build upon this thesis.

2 LITERATURE AND BACKGROUND

2.1.1 CORNERSHOP BACKGROUND

Cornershop started in 2015 by two Chileans and a Swedish that previously worked together in two other startups. It acts as a marketplace for brick-and-mortar stores to have a digital sales channel that is also extremely fast from the moment the purchase is made until it has been delivered.

For earners, the company acts as a source of income with the constant offering of the tasks to fulfill the orders placed through the marketplace, providing an accessible income source for that value the flexibility of working on their schedule.

Cornershop currently operates in 8 countries, over 100 cities, and had a majority stake acquired by Uber in 2019 with the deal finalized in January of 2021.

The recent acquisition makes Cornershop a part of the different ventures acquired by the US based company, which also has Uber groceries that offers a similar platform as Cornershop but where there is little market overlap.

The company started its operation with a dual launch in both *Ciudad de Mexico* and *Santiago*, and initially focused on expanding its offering in the markets of these two countries, with a focus on delivering your weekly groceries with an emphasis on *promise fulfillment*[1] and reputation for a more reliable customer support.

The company currently offers its delivery service in 8 countries and rapidly increased the cities and countries where the service is available during the COVID-19 pandemic. Aside from growing in terms of availability, the participation of non-grocery type stores has also increased. Whereas traditionally the service was almost exclusively associated with the delivery of groceries, currently a sizable portion of its sales comes from smaller specialized stores.

2.1.2 DELIVERY SERVICE DESCRIPTION

As with most other *gig economy*-based platforms, the core of the service provided is based on coordinating a pool of independent contractors, that through the company, receive training on the ins and outs of the operation and then log in via a mobile phone app when it seems most convenient to them.

[1] The service is offered within 1- or 2-hour delivery windows, whereas more traditional retail services in these countries have offered either “within the day” or “am or pm” deliver windows.

The contractors (for the purpose of Cornershop's business called *shoppers*) are free to choose when, where and for how long they are available for jobs to be assigned to them. The only "input" the company has with regards to *shopper availability* is with dynamic incentives for the earnings per job a shopper receives.

Shoppers are offered jobs, which for the context of this thesis will only consist of bundles of orders and their corresponding milestones, which can be accepted or rejected. Given the contractor nature of the business model in which the platform operates, the offer of future jobs cannot be conditioned on the results of accepting or rejecting current ones.

Taking into account a *shopper's* past order acceptance would condition the shopper's livelihood as well as potentially facing legal challenges in some of the markets the company operates in or may wish to operate in.

2.1.3 *RIDE HAILING SIMILARITIES AND THE SOCIAL BENEFIT OF A DELIVERY SERVICE*

The retail personal shopper (RPS), which the shopper matching model forms a part of is rather new. Companies that provide this service started approximately 3 to 4 years after app-based ride hailing services such as Uber and Didi, which are also a recent business proposition (all started following the subprime recession in 2008).

The RPS problem is best described as a variant of the online crowdsourced personal shopper problem described in (Arslan A. M., Agatz, Kroon, & Zuidwijk, 2019), which is also similar to the non-crowdsourced problem described in (Arslan, Agatz, & Klapp, 2019).

The problem is characterized by a set of *N shoppers* that are free to determine their availability and acceptance of offered jobs, a set of M tasks that arrive continuously during the operating window (this can be safely considered a single day for the scope of this thesis). For each consecutive problem, the characteristics of available *shoppers* and jobs are known, however there is no guarantee that they will remain.

The matching aspect is defined as minimizing the costs of assigning *shoppers* to the corresponding tasks such that the allocations comply with business operating constraints (such picking location availability windows, vehicle capacity, etc.) and fulfills as many of the tasks as possible.

For the scope of this thesis the matching tasks are tasks batches rather than tasks themselves (a single task may contain the activities to fulfil multiple orders), which makes the *shopper* aspect of the RPS problem identical to the ad-hoc nature of the personal *shopper* problem considered in (Arslan A. M., Agatz, Kroon, & Zuidwijk, 2019).

The RPS problem has a lot in common with app-based ride hailing services that previously mentioned companies provide. Like app-based ride hailing services, the company's operation relies on a dynamic arrival of resources (orders and *shoppers*). Other companies that operate in the same market, have a business model which primarily assigns *shoppers* to fulfil a schedule resulting from the planification of programed orders, with a smaller online component.

The app-based ride hailing services has precedent in literature as a particular case of the dynamic ride sharing (DRS) model as described by (Agatz, Erera, Savelsbergh, & Wang, 2012), particularly the single rider variant where a matching agency coordinates a two-sided market.

In these app-based two-sided market platforms, the agency or company's sole role is of coordinating customer requesting trips with providers fulfilling these requests[2] without the physical interaction of the coordinating agency.

It is relevant to note that many of the same enabling factors that allow these DRS problems to prosper, allow the RPS businesses to function efficiently. It can be stated that in large part the efficient operation of companies that participate with RPS type services build upon the work done by DRS type services.

As described by (Furuhata, et al., 2013) and (Agatz, Erera, Savelsbergh, & Wang, 2012) these factors are mainly:

- The ease of efficient and fast coordination enabled by the widespread use of internet connected mobile phones with GPS type functionality[3].
- Automated matching platforms.
- User based rating class systems to address trust issues that arise from strangers fulfilling services.
- Dynamic in that they provide solutions with very short notice
- Independent with the coordinating agency rarely[4] physically interacting with either providers or customers

From the factors mentioned, the trust users must place in the provider to offer a reliable and safe service is different for the DRS problem than for the RPS type problems. For companies that provide a DRS platform user trust is expressed as customer willingness to get into a stranger's car.

For companies that provide a service with a RPS model trust is expressed as customers willingness to have a stranger pick and choose groceries as well as time sensitive goods (e.g.: a birthday cake with same day delivery).

How well the service fulfills customers' expectations is, as described by (Furuhata, et al., 2013) and (Agatz, Erera, Savelsbergh, & Wang, 2012) a key enabler of these type of services, with Cornershop's CEO publicly stating that "if people trust us to buy avocados for them, they will trust us to buy groceries for them".

From a matching aspect the companies also may choose to formulate their automated matching algorithms in a similar fashion, with (Agatz, Erera, Savelsbergh, & Wang, 2012) describing the

[2] Some variants of the more traditional car-pooling implementation are also a part of the offering of these services; however, the described service remains the main offering for these companies.
[3] GPS is a US based system, in some markets positioning systems operate using European, Russian, Chinese based systems or a combination of them.
[4] Some in-person interaction usually takes place during onboarding activities with providers, also some companies provide in-person offices for more elaborate problem solving with providers.

variants for DRS as either a *Minimize system wide vehicle distance travelled, Minimize system wide vehicle travel time* or *Maximize the number of participants*.

For Cornershop's case both the company formulation and those proposed in this thesis are formulated as minimizing travel times, although under scenarios where there is a poor order to shopper ratio, the lower amount of arbitrary constraints may yield a higher coverage at an efficiency metrics cost, behaving as maximizing the number of participants.

Another key aspect lies in the *independent* characterization of the service, where providers are not employed by the coordinating agency and there for are free to join or leave the pool of "matchable resources" as they please.

The independence factor makes the matching problems have both continuously arriving and leaving providers and customers, which could be exploited for further efficiency gains by considering the stochastic aspects of how future scenarios may be more efficient than the current available matches.

Many of the same challenges faced by the RPS problem are also faced by the DRS businesses, aside from the previously mentioned trust aspect of the service, they both have a chicken and egg situation. The matching agency must provide enough customer volume to shopper associates for them to have an interest to participate in the service and have associates (shoppers) to provide a reliable service to customers.

Both businesses rely on achieving critical mass in order to attempt to have an efficient solution for both providers and customers, this concept from a pricing point of view is briefly mentioned in (Woodard & Microsoft Research, 2018).

One aspect where the RPS and DRS businesses differ is in that of pricing, particularly for providers as both usually show an up-front estimate to customers and commit to charging that amount unless a change is made in the time between placing the request to it being fulfilled.

RPS formulations sell the convenience of accessing stores that customers could visit personally, there for are inclined to offer the same prices as these stores do in person (revenue is usually generated by a fee in agreement with the store).

For RPS operations, pricing aspects are often more closely associated with online meal delivery service platforms, which also builds upon many of the same DRS aspects as previously mentioned. DRS and RPS problems alike usually rely on an agreement between a merchant and the coordinating agency for their revenue and have either a catalog or event-based compensation model for providers.

For simplicity, the same problem description can be reasonably extended to the online meal delivery service, where the main differences with the RSP problem reside in the importance of the picking portion of the job fulfilled by the provider.

From a modeling point of view, the RPS problem can be viewed as a special case of the generalized pickup and delivery problem (GPDP), with the GPDP understood as the matching problem to fulfill a set of requests that require a delivery, usually with time windows, that must be picked up at a defined location which may be different from the location of the agent that fulfils the request (Savelsbergh & Sol, 1995).

The version of the GPDP that the RPS problem solves is that described in (Parragh, Doerner, & Hartl, 2007), which is essentially an open-ended Travelling Salesman Problem (TSP), unlike the problem of the GPDP the formulation of the RPS should also consider picking aspects within the costs of the objective function. For the purposes of this thesis the picking aspects are captured by the handicap component of the objective function as described in 3.2.4 .

Commonly, the RSP problem also shares time window constraints and costs with the GPDP variants, these consider a hard time window for store availability, when customers are promised a delivery window a soft time window for the delivery.

The novel aspect of the RPS problem that is not considered in the GPDP problem, is the addition of the picking activity. Assuming a heterogeneous set of shoppers, the performance of the operation will be affected with how good a match are shopper characteristics for orders characteristics, where more experienced shoppers are expected to perform better (less mistakes and faster picking) for large or more complex orders.

The impact the picking activity has on the formulation versus a single heterogeneous Pickup and Delivery Problem (PDP) is seen in costs related to picking in addition to travel time costs. Merchant requirements could also introduce additional constraints to the formulation, such as requiring credentials to access a merchant location.

From the operation of cornershop, examples of additional constraints form a base formulation include merchant credentials required for shoppers to enter some stores, store shopper occupancy limits, alcohol shopping eligibility, etc. For a more self-explanatory definition, this variant of the GPDP can be referred to as the *pickup activity and delivery problem* (PADP).

Agents for the dispatch PADP at cornershop are shoppers, however, unlike previously described for the RPS problem, the jobs that require picking, pickup and delivery are order batches and not orders themselves. For this thesis, order batches are referred to as an *assignable*, which is a request for the activities to fulfill one or more orders.

The company operates with *shopper* matching and order batching problems defined as separate problems, the objects to match for *shopper* matching are the batches generated by the batching component, which may contain a single order or multiple if a batch was found.

For this thesis the batching of orders into these *assignables* is considered a separate problem, the matching job becomes how to find the most efficient allocation of shoppers to these *assignable* jobs regardless of how efficient these order batches may be.

2.1.4 *ENVIRONMENTAL IMPACT ON EFFICIENT SHARED VEHICLE DELIVERY SERVICES*

The environmental impact from logistics in delivering goods has garnered increasingly more attention in recent years, the majority of retail operation emissions is from CO_2 and equivalent emissions.

Albeit focused on more traditional warehouse distribution logistic models, the use of distributed logistics and the use of shared transportation vehicles does achieve a reduction in vehicle miles travelled (VMT) in comparison to personal vehicles fulfilling these same trips (Wygonik & Goodchild, 2012).

Because the emissions for a comparable car are directly related to the travelled distance of such cars, a reduction in travelled distances also results in the corresponding reduction in emissions, thereby improving the environmental footprint of retail deliveries.

The key principles that make shared use vehicles a less pollutant solution still hold true for the case of using smaller personal vehicles with its corresponding smaller products capacity constraint, particularly when one considers that given the compensation structure for shoppers is not vehicle dependent[5], makes it less likely that shoppers utilize vehicles on the less efficient side of the spectrum.

It is there for a fair extrapolation that reductions in travelled times achieved by a more efficient matching of *shoppers* to *assignables*, should also result in a reduction in the greenhouse gas emissions from the company's operation. This means that efficient allocation of shoppers not only has a business motivation, but also a social well-being one. (Yu, Tang, Li, Sun, & Wang, 2016)

2.2 PROBLEM DESCRIPTION

As a delivery platform, Cornershop has managed to stand out by prioritizing customer satisfaction, which is reflected in offering the delivery of an order with a promised narrower time windows, order trouble shooting and a more dependable customer support. This at a higher level than what companies that traditionally operate with a sizable brick-and-mortar stores, many of which are Cornershops' partners.

As a platform-based service, the company relies on a pool of *shoppers*, that sign into the platform and are offered the job of picking and delivering an order that a customer placed with their account (usually through a cellphone app, though a web portal is also available).

The company must allocate these jobs in such a way that costs are reduced, while maintaining metrics that directly impact customer perception. Metrics that measure customer's service quality perception are delivering the order within the promised time window and managing to find and deliver the requested products with a suitable replacement when not in stock, amongst others.

[5] Shopper compensation plans for Cornershop are vehicle type dependent (cars, motorcycles, bicycles walkers…), but are uniform within a given type (a shopper with a gas guzzling H1 Hummer is compensated with the same plan as if he were to drive a Prius hybrid)

While the company must seek to allocate jobs in a way that would minimize its costs while keeping within a certain "quality threshold", it must also take into account the value it provides to both customers and shoppers alike when allocating a match. Value for *shoppers* is pursued by trying to avoid the accumulation of orders in a comparatively small set of "*super shoppers*", as this would hamper the company's ability to grow and welcome new *shoppers* to the platform.

Currently the company uses a greedy matching approach that filters candidate matches that comply with business policies, then are sorted within indicators of efficiency (such as estimated total elapsed time), *shopper* allocation fairness and penalties for allocations that have a higher chance of resulting in a poor customer experience.

The heuristic, described in more detail in 3.1, runs in parallel for each zone with a discrete time interval (30 seconds). The problem is solved for a scope of all *assignables* with the delivery location contained in a partition of the geo-polygon where the service is available.

For this thesis the geo-polygons are referred to as operating zones. The availability of the service in a city could be represented by a single zone or split into chunks. Each of these chunks or zones is treated as an independent problem with no overlap.

To solve potential conflicts in the *shopper* allocations for different zones in the same city, a tie-breaking heuristic is used to determine the allocation of resources. The heuristic enforces the constraint that a single resource cannot be allocated to multiple jobs simultaneously, this aspect refers more to the engineering aspects of the implementation and will not be considered in this thesis.

For the context of this thesis, it will be assumed that only a single *shopper* fulfills an order (within the operation *shoppers* could be allocated to just a fraction of the order workflow process due to a contingency), however the matching is done not with orders and *shoppers* and stores, but rather with packs of orders that may contain a single order or a set of pre batched orders, where some business constraints only apply to a pack that is not a single order. For the purposes of this thesis, the object of order batches is called *assignable*.

The method used to batch these orders is responsibility of a separate model and won't be considered within the scope of this thesis, although it could be feasible to run a model that addresses both matching and batching tasks, the focus of this work is to evaluate the method for solving the matching problem and not to change the problem itself.

2.3 INDUSTRY REFERENCES

2.3.1 GENERALIZED ASSIGNMENT PROBLEM

Assignment and matching problems are well studied within the operations research community, they deal with the question of how to assign or allocate a set of *m* agents/workers to *n* jobs. The generalized assignment problem (GAP) is defined using both knapsack (items and weights) and scheduling terminology (agents and jobs). For this thesis the scheduling terminology is used. (Öncan, 2007)

The GAP has seen applications in several businesses from logistics (Baker & Sheasby, 1999; Srinivasan & Thompson, 1973), manufacturing (Higgins, 1999; LeBlanc, Shtub, & Anandalingam, 1999), energy management (Yu & Prasanna, 2003), project management (Drexl, 1991), telecommunications (Bressoud, Rastogi, & Smith, 2003), etc.

Even though the problem is solved in a recurrent fashion, within the scope of this thesis no aspects of an online or stochastic problem are present in the approach used by the company. All stochastic components are taken at face value as deterministic inputs and no interaction between matches of previous runs of the problem are used.

The GAP is one a combinatorial problem and is known to be NP-hard (Sahni & Gonzalez, 1976), luckily being such a well-studied problem, there have been many proposed algorithms for solving GAP's, including exact approaches, heuristics and metaheuristics.

Commonly used approaches within the industry are based on the Hungarian algorithm or some form of solving the Lagrange relaxation of the problem with some form of feasibility heuristic. This last approach benefits from fast implementations of linear solvers if the feasibility heuristic is not compute heavy. (Öncan, 2007)

Generic bi-partite weighted matching formulation with set cardinality approach

$$\min_{x} \sum_{i \in I, j \in J} x_{i,j} \times c_{i,j} \quad (1)$$

$$s.t. \sum_{i,j} x_{i,j} \geq \min(|I|, |J|) \quad (2)$$

$$\mathbf{x_{\{i,j\}}} = \begin{cases} \mathbf{1} \ \boldsymbol{if\ i\ is\ matched\ with\ j} \\ \mathbf{0} \ \sim \end{cases}$$

2.3.2 DEFERRED ACCEPTANCE

Up to this point the actions that happen after a matching has been found have not been discussed. A fundamental part of the business model of gig-economy services lies with the fact the workforce that provides whatever service the company offers (groceries and retail delivery in case of Cornershop) is done by a flexible and external workforce.

With a subcontractor workforce there are strict legal limits for how the company can interact with earners. Due to the nature of the operation, there will always be freedom for contractors to reject an allocated matching. Rejected allocations poses a great challenge since these orders must be re-allocated at a later opportunity, this means waiting for an acceptance whilst the chance of the order arriving late increases.

A potential formulation that could result in a higher chance of "problem jobs"[6] being accepted by the available contractor workforce would be treating the matching problem of each instance as a deferred acceptance problem. The deferred acceptance approach is not too distant from more well-known applications of this formulation in auctions and student school assignment used in real world applications in Boston. (Abdulkadiroglu, Pathak, Roth, & Sonmez, 2006)

The deferred acceptance problem is that which generates matches such that the solution contains no unstable matches, where a match between two *assignables* A, B and two *shoppers* α, β is said to be an unstable match if α, β who are assigned to A, B respectively even though α prefers B and β prefers A. (Gale & Shapley, 1962).

A deferred acceptance formulation should yield an improvement in the acceptance rate of *assignables* from *shoppers*, however this may not translate to a practical implementation as it would require the company being able to accurately model each individual *shopper's* opportunity cost and preferences.

[6] Problem Jobs from a contractor point of view are those where a perception of low earnings or that may hamper their ability to obtain more jobs upon completion of the current one.

The current approach used by the company already yields high acceptance rates as incentives for both parties (company and *shoppers*) are already well aligned with the current formulation. This alignment makes it less likely that potential gains from a marginally higher acceptance rate would be worth it over a purely cost minimization approach.

The matching scenario for a gig-economy workforce also introduces a high degree of information asymmetry between agents and coordinator, which limits the effectiveness of a sophisticated response strategy by the shoppers, which makes one of what can be considered the motivating principals to proposing a deferred acceptance formulation less relevant for the overall objective of the matching problem proposed for Cornershop's *shopper* allocations.

3 PROBLEM

For allocation purposes, the daily operation is partitioned into 30 second intervals that are solved as isolated problems. A recurring process checks which order batches must be assigned within the next chunk and which *shoppers* and stores locations are available for those order batches. Each of the allocation chunks is defined as an instance of the matching problem.

Each of the described allocation problems runs independently for an *operating zone*, which is a geo-fenced partition of a city where the service is provided (the partition could encompass the whole city for smaller cities). For the purpose of thesis only the scope of a single of these *operating zones* is considered for each scenario, interaction between zones and aggregate results across zones are not modeled.

The problem of shopper-assignable-store assignment is a tri-partite allocation problem, where one assigns a set of available shoppers to offered order batches (orders are treated as order batches called *assignables*, these may contain a single order or multiple orders) to a set of *branches*, which are the physical stores at which a particular assignable can be picked.

For each instance the available data is:

- The set of *assignables* with characteristics (size, delivery location, items, etc.)
- The set of *shoppers* with metadata (transportation, location, past performance, etc.)
- The set of *branches* with metadata (open hours, location, current and max capacities, etc.)
- Pre calculated costs (travel times, business skew, travel distances, etc.)
- Fixed instance parameters (transportation limits, fixed costs)

3.1 COMPANY PIPELINE MODEL FORMULATION

The company model, which for this thesis is the *pipeline model*, will search iteratively for the available resources within each instance. The *assingables* within an instance are sorted in a list, which defines the order in which the matches will be searched. Once an allocation has been found for an *assignable*, both it and its matched *shopper* are then removed from options for future matches, hence the greedy nature of the company model.

For each *assignable*, the *pipeline model* then selects the most efficient shopper with two steps:

- **Preparation phase**
- **Selection phase**

The **preparation phase** will filter allocation candidates that don't comply with business feasibility criterions and then add data required for the selection phase to the remaining match candidates. The business feasibility constraints can be described as performing the following checks:

- Shopper transportation type compatibility (e.g.: fitting inside a shopper backpack)
- Assignable content that may require special care (e.g.: frozen items)
- Legal order requirements (e.g.: Alcohol orders with underaged shoppers)
- Shopper limits (e.g.: max order weight accepted by shopper)

- Assignable with “special” considerations (e.g.: new customers with experienced shoppers)

From the remaining match candidates with its corresponding data, the **selection phase** will sort the business feasible list in the following fashion:

- Sort for *min allocation cost*
- Of the min cost + tolerance candidates, sort for longest shopper idle time

The main criteria for *shopper* selection, is an allocation that has the minimum possible allocation cost, which is primarily the travel times to and from the partner store where the *assignable* is to be fulfilled. The *allocation cost* also considers a skew cost that reflects business preferences.

The min cost plus tolerance workflow works as a tie breaker heuristic for *shopper* and store combinations that have the same allocation cost (Total Time), the nuance for this heuristic is that any combination that is at most “tolerance away” from the most efficient allocation candidate is also considered as a tied candidate.

The tie breaking metric used is the amount of time *shoppers* have spent idle since the last completed task.

From the sorted list, the first *shopper-branch* combination is then selected as the match for the assignable. This selection process is repeated for all *assignables* or until all the instance shoppers have been assigned.

The allocation cost is named **Total time**, for an *assignable* a, shopper *s* and *branch* b it is defined as:

$$total\_time_{a,s,b} = Shopper\_to\_branch\_time_{s,b} + branch\_to\_delivery\_time_{a,s,b} + handicaps_{a,s,b}$$

where the travel times are estimations obtained from an external provider and the *handicaps* are a skew for a given match candidate according to business preferences, examples of these are:

- New shoppers being preferred small orders and are assigned a negative handicap when orders contain less than a set number of SKUs
- Experienced shoppers being preferred for first time buyers, for first time buyers a negative handicap is added for *shoppers* that have a positive customer score
- Bike shoppers being preferred for small orders, when an *assignable* has less than a set number of SKUs a negative handicap is added

The selection *tolerance*, used for the tie breaking component, is a fixed value in the same units as for the *total time* cost, it allows for all match candidates that are at most *tolerance* away from the ideal candidate to also be considered.

3.1.1.1.1 Pipeline model – pseudo code

3.1.1.1.1.1 Inputs:

- List of instance *assignable*
- List of *shopper-branch* (store location) tuples

Goal:

- **Optimal ids for *shopper-branch* allocation with *assignables***

Algorithm 1: Pipeline model – pseudo code

Require: List of **Assignables** $A_1 \dots A_n$
Require: List **of shopper-branch** allocation candidates $SB_1 \dots SB_n$
Return: Optimal ids for (*Shopper, Assignable, Branch* tuples)

for $assignable$ **in** $assignable\_list$:

for allocation_candidate **in** $shopper\_branch\_list$:
Check transportation requirements
Check for *assignable* contents that require special care
Check legal feasibility requirements
Check *shopper* limits
Check *assignable* limits
Set *handicaps* for allocation-candidate
End for
sort allocation-candidate list **by** *total time cost*

for *allocation-candidate* **in** sorted-allocation-candidates:
If $allocation\_candidate\_cost - \min(sorted\_allocation\_candidate\_cost) \geq threshold$:
Remove *allocation_candidate*
End for
Sort *allocation_candidate* **by** shopper_idle_time:

$\min(sorted\_allocation\_candidates\ by\ idle\_time)$

End for

3.2 MATHEMATICAL OPTIMIZATION

For ease of readability the problem formulation is presented in the following segment. The explanation of why the problem is modeled with the proposed formulation is explained after the model.

3.2.1 SETS

The sets for each instance are defined as follows:

- Shoppers ($s \in S$)
- Assignables ($a \in A$)
- Store locations ($b \in B$)

3.2.1.1 Model resources

The available information for the model **preparation phase** is functionally identical to that of the *pipeline model*. The selection phase differs in that the inputs are:

- A list of business feasible *Shopper, Assignable, Store location* allocation candidates
- The costs associated with each feasible allocation candidate (travel times and handicaps)

3.2.1.1.1 Decision variables

Matched found

$$\mathbf{x_{s,a,b}} = \begin{cases} \mathbf{1} \ \boldsymbol{if\ shopper\ s\ is\ allocated\ to\ assignable\ a\ at\ store\ b} \\ \mathbf{0} \sim \end{cases}$$

Non allocation

$$\boldsymbol{y_a} = \begin{cases} \mathbf{1}\ \boldsymbol{if\ assignable\ a\ is\ not\ matched} \\ \mathbf{0} \sim \end{cases}$$

$$\boldsymbol{X_{\{s,a,b\}}, Y_{\{a\}}} \in \mathbf{Binary}$$

3.2.1.1.2 Constraints

store capacity constraint

$$\sum_{s \in S, a \in A} x_{s,a,b} + \text{current\_capacity}_b \leq store\_capacity_b \quad \forall b \in B \tag{3}$$

Non allocation activation

$$1 - \sum_{s \in S, b \in B} x_{s,a,b} \leq y_a \quad \forall a \in A \tag{4}$$

Assign at most one assignable per shopper

$$\sum_{a \in A, b \in B} x_{s,a,b} \leq 1 \quad \forall s \in S \tag{5}$$

Assign at most one shopper per assignable

$$\sum_{s \in S, b \in B} x_{s,a,b} \leq 1 \quad \forall a \in A \tag{6}$$

3.2.1.1.3 Objective function

$$\min_{s,a,b} \sum_{s \in S, a \in A, b \in B} x_{s,a,b} \times total\_time_{s,a,b} + \sum_{a \in A} y_a \times non\_allocation\_cost_a \tag{7}$$

3.2.2 *DECISION VARIABLES*

The *pipeline model* allocates shoppers in an approach that is akin to a bi-partite matching formulation with a set size constraint, this approach is similar to a mathematical optimization model shown with (1) and (2).

This approach evidently can be replicated in a mathematical optimization approach, as evidenced by the approach of using (2). Forcing allocations based on cardinality does not adequately capture the business decision that a match candidate may be feasible, but not desirable.

The proposed formulation introduces a penalty for not allocating an *assignable*, this allows the company to explicitly set a threshold for how inefficient an allocation is allowed to be a preferable one or if it will defer the allocation for the next instance. It is worth noting that with sufficiently large penalty costs, this approach is equivalent to the set cardinality approach illustrated with (1) and (2).

A benefit of having a penalty for "incentivizing" allocations, instead of a requirement based on the size of a set, is that this approach allows the user to set a threshold for what is considered a desirable match based on a specific set of indices and its associated cost, as opposed to having the decision be made up-front.

A difficulty introduced by this approach is that it makes the solved objective function cost of the proposed models not directly comparable with the solutions for the *pipeline model*. The efficiency of allocations is evaluated only considering the allocation cost component, however this is does not properly take into account the tradeoff with the allocation coverage, which is heavily impacted by the value of the non-allocation cost.

The results between allocation coverage and efficiency metrics are discussed in further detail in chapters 5 and 6 of this thesis.

For simplicity, for this thesis the cost introduced for not allocating an *assignable* is exclusively dependent on *assingables*, however this concept can easily be extended to the *shoppers* and *branch* sets for implementations out of scope for this thesis.

3.2.3 *CONSTRAINTS*

The proposed mathematical optimization models are comprised of two types of constraints:

- Constraints that are active or not regardless of the value of decision variables, such as blocking under aged *shoppers* being allocated an *assignable* with alcohol content. These constraints will be referred to as **implicit constraints**
- Constraints which depend on the value of decision variables, such as controlling for store occupancy, which must consider how many *shoppers* are being sent to a store. These constraints will be referred to as **allocation constraints**

Since **implicit** constraints aren't affected by the values of decision variables, these are applied in pre-processing by filtering candidate matches for a given instance, as they are independent of the

variables themselves. The implementation used is a sparse representation of indices for the decision variables created in the model to only the ones that meet these constraints.

As for **allocation constraints,** are dependent on the allocations for a given instance, these constraints are modeled within the formulation of the problem for an instance. These constraints are both for the correct functionality of the model as well as business derived constraints.

3.2.3.1 Implicit constraints

Since these constraints are independent on what values decision variables take, the most efficient way to add them to the formulation is in pre-processing, as adding them in the declaration of the model with python APIs results in poor performance.

Generating the feasible allocation candidate tuples is at least an order of magnitude faster than modeling these constraints by declaring them with a mathematical representation in solver compiler.

For medium to large instances of the proposed base model, the compilation time is reduced from over 5 minutes when declared in the model to less than 5 seconds of compile time, whilst the impact on additional pre-processing time is measured in seconds and can be easily parallelized, which can't be achieved using python solver interfaces.

Applying the set of **implicit constraints** to an instance is as mentioned an *embarrassingly parallel workload*[7]. For the purpose of this thesis these constraints are parallelized on a *shopper* basis, meaning that building feasible match candidates is a separate job for each shopper.

To reduce parallelization overheads, the set of shoppers is split into equally sized batches of *shoppers* as many cores the machine has (32 threads for this thesis). The application of the parallel approach manages to reduce by almost half the compilation time for larger instances used in this thesis, however it does result in longer compute times for small instances.

In aggregate, the pre-processing workload yields a list of match candidates that can't be discarded without knowing the state of decision variables of the instance at optimality.

The type of business constraints applied as **implicit constraints** can be described as one of the following checks:

- Shopper transportation type feasibility
- Assignable contents that require special transportation care
- Legal requirements that Cornershop must comply for *shopper* well being
- Shopper declared limits (shoppers can set limits for what they are willing to carry)
- Assignable type that requires particular *shopper* features

[7] Embarrassingly parallel is a workload that can be easily divided into components that can be executed concurrently. (The Art of Multiprocessor Programming, Revised Reprint; Maurice Herlihy, Nir Shavit)

3.2.3.2 Allocation constraints

The main business constraint that depends on how one allocates *shoppers* amongst *assignables* is that of the store capacity limit, where every store location can have a maximum allowed occupancy associated with online orders, this constraint has become particularly relevant with the sanitary measures implemented by different governments due to COVID-19 guidelines.

Due to the way in which the *pipeline model* is implemented, checking for total estimated merchant occupancy is impractical, as it would require redefining pre-calculated values in each of the *shopper* searches for all *assignables*. Due to this limitation, the branch occupancy constraint is implemented as a *handicap* (business skew) that is inversely proportional to the remaining capacity of a given merchant branch.

To avoid proposing an approach that would be a heuristic of a heuristic or complicating unnecessarily the problem formulation, the constraint will be implemented directly as an upper capacity limit for each store location, since the mathematical optimization approach does not have the same limitations as the *pipeline model* does, this more closely resembles the actual problem the business needs to solve. This constraint is modeled as (3).

3.2.4 OBJECTIVE

As mentioned in the description of the *pipeline model*, the metric to minimize is the allocation cost, which for this thesis is referred to as: **Total time.** For the optimal value of total time, an additional step takes place to take shopper idle times into account. The proposed formulation is explicitly considering the efficiency aspect of minimizing the allocation cost, the "fairness" criterion is addressed with the **fairness model** explained in 4.3.

As mentioned in 3**.2.2**, the proposed formulation adds the concept of explicitly not allocating an *assignable*, which must also be taken into account for the objective function of the formulation. The objective function of the proposed formulation is explained as the minimization of sum of all allocation costs and the costs of not allocating an order (7).

Total time cost is defined as:

$$Total\_time_{s,a,b} = shopper\_to\_branch\_travel\_time_{s,b} + branch\_to\_customers\_travel\_time_{s,a,b} + handicaps_{s,a,b} \quad (8)$$

Travel times of the Total time cost are self-explanatory, for the "branch to customer" component the sum of all legs in the trip are considered, so for *assignables* with multiple orders it is the sum of the first leg between the branch and first customer and all following legs between customer delivery locations.

The handicaps component is the sum of all business skews for that particular *shopper-assignable-branch* match candidate. The scope of what these businesses preferences are is covered in chapter 3.1.

It is important to highlight that a skew or handicap may be both positive or negative, which could result in the Total time cost for a match candidate being negative if travel times are small and there is a large handicap cost associated with the match candidate.

For instance, if a first-time buyer has a delivery address near the store, and a *shopper* is found waiting at the door, the travel time components would be close to zero for shopper to branch, and small for branch to customer, let it be 8 minutes branch to customer expected time. If the company has defined the new customer *handicap* as a 15-minute benefit for eligible *shoppers*, the **total time** value for that allocation is going to be -7 minutes.

4 PROPOSAL EVALUATION

The operational performance of the mathematical optimization approach is implemented in a local machine with scenarios that are derived from the company's historical data. The scenarios used in this thesis are not a carbon copy of observed scenarios but are based on operational information. The details of the differences between these "historical runs" and the ones used for evaluating the optimization model are discussed in this chapter.

4.1 SCENARIO SOURCES

Due to contract limitations, not all the data used for company version of the model is stored, also some fields used during the runs of the company model (a.k.a. the "*pipeline model*") are discarded after use, whilst others are stored in dynamic datastores[8], meaning that any updates to a that information overwrites the existing values.

Given the difficulties that arise when trying to exactly replicate the scenarios observed in during the operation of the *pipeline model*, the aim is to generate new scenarios that are plausible but may contain some differences to the information used for matching initially. The testing scenarios or "sequences" are built using the following sources:

- Logs of sequences faced by the *pipeline model*
- Dynamic data stores with the most recent values
- Test version of data that provided by external suppliers

Log data

Data contained within these logs is a recording of data used for the production version of the matcher, as previously mentioned it does not contain all the data used during a run of the matching algorithm.

Dynamic data stores

Some fields used within the match optimization process (in the *pipeline model*), are stored in a way that overwrites past values when a change is introduced. For this thesis this implies that for these fields will be that of when the data was queried, which may differ from the value of the corresponding field when the log data was created.

Even though the value for these "overwritten" fields may differ from the original values, the format remains the same as that when the instance used as basis was created.

A small portion of the information housed in these dynamic data stores does record each update separately, for these fields the value used was that which was active when the instance logs were created.

[8] A data store is a form of persistent storage, where the structure is not specified. This may include a relational database, structured and un-structured log storage, blob storage amongst others.

External supplier information

Data from external providers is often legally limited as for time span which it can be stored within company data stores, these limits make using the exact same source that the saved logs used unpractical or illegal, however one can legally use a static version of the data consumed exclusively for analysis.

For matching purposes, the only relevant information provided by an external supplier is the elapsed travel time between points within a city, this information is generated by Cornershop using geospatial base files that change daily and due to contract compliance can't remain in company data stores for more than 48 hours. The base files mentioned are provided by an external supplier which limits the use of these.

For the purposes of this thesis a static version of that geospatial data package is used, which is functionally the same as the one used for production, however it may yield different values as the ones for the date of the logs. The static version of these geospatial files can be stored by the company.

4.2 MODEL RUN SETUP

The production version used by the company runs throughout the day with a constant frequency (for this thesis 30 seconds). For each run or "instance" a snapshot is taken of the available resources used for matching (connected *shoppers*, *assignables* and store locations), the way the resources are updated is out of the scope of the allocations model, its only job is defining the matches given a set of available resources.

For the local version of these runs, a sequence of these *instances* is pre-loaded to the machine derived from logs before any optimization, additional information is then loaded to these files and travel time information is re-generated using the geospatial files. These values are precalculated for all possible scenarios faced during the run of the sequence of instances imitating the way information would be available for a production version.

Once a run information is loaded a runner then runs, inherits, and records the solutions for the specified matching models for every instance within the loaded sequence. Once all the runs are complete, the output is the matching recording with its corresponding metadata.

All models have the same starting instance. Instance updates derived from log data is also the same for all models, however unlike the first instance, instances between models will have differences due to the impact of earlier instance matches. For a given instance i_n, where $n \neq 0$ the data loaded will be the log data for instance i_n plus, the results impact from all instances before instance n.

The "inheritance" between each instance works by adding or subtracting information to a following instance depending on the matching decisions of the previous one. The inheritance decisions considered are:

- Matched *assignables* are flagged as such and are eliminated from any future instance if present.
- *Assignables* which are not allocated within a given instance are added to the following one
- *Shoppers* allocated to an assignable are flagged as such and are no longer available for subsequent instances
- *Shoppers* not allocated in an instance are considered as available *shoppers* for following instances. The same *shopper* information is used for future instances unless a future instance contains an update for the information of that *shopper*. If an update is found, the updated information is used.
- If a store location (a.k.a. store branch) is available for an instance, then it is made available for all the following instances, with the caveat that its capacity is updated dependent on the cumulative amount of *assignables* allocated to that branch for the previous instance.
- If log data for an instance contains new updated information for a *branch,* then the updated information is used except for the value of branch current occupancy, where the max between the updated and the state variable of occupancy for that branch

The only bit of information generated during the runs of each instance is a counter for the amount of tries it takes to find a match for an assignable within the run. This functionality is implemented within the inheritance between instances e.g.: if an *assignable* is matched on the first try, it will have an allocation count of 1, if no match was found twice and then allocated the allocation tries will be 3.

4.3 MODEL VERSIONS

For this thesis a mathematical optimization model is benchmarked against a local implementation of the model used in production by the company (*pipeline model*). Three mathematical optimization versions are also benchmarked alongside the proposed model, these are the same model as proposed in chapter 3 using different model parameters.

The pipeline model is implemented using a containerized version of the production replica and is completely python 3.8 . All the mathematical optimization models are implemented using python 3.8 for preprocessing and handling all the data, as well as a commercial solver (Gurobi 9.1) with its proprietary python compiler.

From now on, each model will be referred to as:

- **Optimal allocation model:** The original mathematical optimization proposed
- **Threshold model:** A heuristic version of the *optimal allocation model* with a lower value for the cost of not allocating an order, the "threshold" value is set as the 90^{th} percentile of the allocation costs for a given week.
- **Fairness model:** A heuristic of the *optimal allocation model* that adds a negative cost to *shoppers* depending on the amount of time each *shopper* was assigned his or her last task up to a value of 30 mins

- **Fairness and threshold mode:** A heuristic version of the *optimal allocation model* with both previously mentioned heuristics working simultaneously.
- **Pipeline model:** This is a local and containerized version of the model used by Cornershop.

4.3.1 OPTIMAL ALLOCATION MODEL

This model is the base proposed mathematical optimization proposal. This model is defined by equations (3) to (7) of chapter 3.2. In order to replicate the behavior of the pipeline model, the cost for *non_allocation* associated with variable y_a in equation (7) is set to an extremely large value (300 minutes for the results shown).

4.3.2 THRESHOLD MODEL

This model is identical to the *optimal allocation model* except for the non-allocation cost in the objective function (7), which is set to a lower value. For this model the non-allocation cost is set as the 90th percentile of the historic total time cost for orders for a given week in the company for the studied zones.

In a deployment of this version of the model, this historical data derived threshold (or benchmark) should be recalculated to tune the performance of the operation as it shifts due to seasonality and some events that trigger large differences in the operation (such as Mother's Day).

The goal of trying out a lower non allocation cost is avoiding the allocation of matches with an unusually large total time (matching cost). The deferral of allocating these orders implicitly contains the hope that a lower cost match candidate would be available for that assignable in future instances.

The downside of using this technique for achieving lower match costs is in longer elapsed times for finding suitable *shoppers* for orders, which could result in orders arriving late more often or arriving late by a greater time difference than that achieved by the *optimal allocation* model.

4.3.3 FAIRNESS MODEL

This model is identical to the *optimal allocation model* except for the values of the *handicaps* component for each *assignable*. For this model a skew is introduced for match candidates, the skew is a benefit for shoppers that have had longer idle time[9].

The size of the introduced skew is the max between 30 minutes and the idle minutes of the shopper for that match candidate.

The goal for this variant lies in assessing the performance of potential solution to add the tolerance aspect discussed in 3.1 to the mathematical model, this is a relevant feature as it allows the company to provide a more interesting platform for markets where order volume is lower.

The implementation of the *pipeline model* does not show a large tradeoff between having the mentioned *fairness heuristic* and a *pipeline model* implementation that solely focuses on reducing

[9] *Shopper* idle time is considered as the time since last completed task that day or time since login if no tasks were completed that day up to that point.

matching cost. In order to consider this variant successful, the matching cost difference between this model and that of the *threshold model* or the *optimal allocation model* should be small.

4.3.4 FAIRNESS AND THRESHOLD MODEL

This model implements both the features discussed for the *threshold model* and the *fairness model* simultaneously. The goal for this version is to assess the fair heuristic proposed for the *fairness model* with a performance enchantment discussed for the *threshold model*.

4.3.5 PIPELINE MODEL

This is the local implementation of the company heuristic discussed in chapter 3.1. The implementation of this model is derived directly from the company code base, with adaptations to accept data format for instances in this thesis, as these are built slight differently of how data is handled in the company code base.

To avoid replicating portions of the complexity that requires the deployment of the company code base, some aspects were emulated. The emulated portions of the mode maintain functionality though these short cuts negatively affect code running time performance.

The use of these emulated deployment components only affects code runtime performance, the decisions are operationally identical to that the company obtains for the same instances.

For each version runs independently of each other, and face the same starting scenarios and updates, however the data for a given instance may be different based on the matches found on previous ones for that particular model.

5 RESULTS

5.1 RUN SCOPE

5.1.1 RESOURCES SCOPE

As previously stated, the problem of *shopper* allocation at Cornershop is run in parallel for each zone, specifically the problem is divided into the resources (*assignables*, *shoppers* and stores) located within operation zone, which is a polygon that encircles a portion of a city where the shopping service is provided.

Each of these zones (referred to as operating zones) can represent the operation of an entire city or be a subdivision of the entire area where the service is provided. For cities that are divided into more than one zone, the polygons of each zone do not overlap, the whole operating area for a city can be viewed as the unary union[10] of all the operating zones for that city (a representation of these partitions can be seen in 5.5).

Since the scope of each problem is that which is encompassed within an operating zone, and the intersection of these is null, all these problems are independent of each other. The implementation of these operating zones could be viewed as solving the problem for the whole city with the added constraint that all resources assigned intersect with the polygon of a given zone, however they are modeled as independent problems as is the implementation of the company.

5.1.2 RESULTS EVALUATION ENVIRONMENT

The results shown in this thesis were obtained using a personal computer with:

- 16 physical Zen 2 AMD cores (using a Ryzen 5950x, 32 logical cores)
- 64 GBs of RAM
- The OS used is a virtualized Ubuntu 20.04
- The code is developed in Python 3.8 using Gurobi python for compiler
- Uses Gurobi 9.1 for solver
- The *pipeline model* used is deployed locally in a docker container

The implementation used for this thesis focuses on functionality rather than performance, some performance is sacrificed to allow for an easier implementation, this makes running time an aspect that can't be reliably compared with a "deployable" version of the model.

The runtime for the mathematical optimization models proposed are well below the required 30 second interval set by the company, which would allow the implementation of any of the proposed models from a performance standpoint, however it requires a more elaborate implementation than that used in this thesis.

[10] GIS unary union is the polygon that is defined by the outline that covers all the joined polygons. For the case of the matching problem, it can be viewed as redefining a subset of problems into a larger one defined by all the distinct resources available within all the zones that define a city's service area.

The performance aspects of the proposed model in a production ready[11] version was tested in company resources in parallel to this thesis, although it is not part of the scope of this thesis, it proves that the compute performance of the proposed model is feasible for the company's operation requirements.

5.1.3 LENGTH SCOPE

As previously mentioned, the matching process is split into discrete intervals that are triggered every 30 seconds. For each interval a recurring process will check what resources (*assignables, shoppers,* stores) are available and then build an instance with these *shoppers*.

For the scope of this thesis, a single matching problem is referred to as an **instance**, a sequence of consecutive instances is referred to as a **run**. The length of a run is 30 minutes of real time operation (60 consecutive instances).

For the purposes of this thesis, the interactions between instances will not be considered unless explicitly stated otherwise, shopper acceptance is also modeled as a certainty for simplicity.

Shopper response to the offered *assignables* is also not modeled, meaning that *shoppers* are not considered to change their behavior depending on what matches are offered. *Shoppers* are also considered as a onetime use resource, meaning that once a *shopper* is allocated, the *shopper* is no longer considered as available for future instances.

The length of runs is chosen so that both it is reasonable to assume that *shoppers* can be treated as a one-time use resource and have a large number of consecutive instances for making it easier to detect any dynamic effects on the **run** results.

Unless explicitly stated, the results shown are for the aggregate allocations of the mentioned period (1 instance for single **instance** runs and 60 consecutive instances for **run** results). For the aggregate results the metrics shown are that of the instance when the *assignable* was matched, regardless of it being the first or n^{th} instance in which the *assignable* was available for a match.

5.1.4 SIZE OF INSTANCES

The performance difference between the mathematical optimization models and that of the *pipeline model* is expected to be larger for instances with a larger volume, as the inefficiencies of the *pipeline model* would have less opportunities to affect the results for instances with a smaller *assignable* or *shopper* count.

For larger instances it is expected that the greedy component of the *pipeline model* is expected to "ignore" more options than the case for small instances, when it is a possibility that the evaluated match candidates are the same for all models.

[11] A production environment is the version of code that is used by the company in its operation and requires the highest level of redundancy, performance and system stability of the code environments used.

In order to show the more relevant results for the operation, in which there would be a greater justification for switching algorithm, the results for this thesis focus on larger instances solved by the company.

Since the results for small instances are expected to be similar for the proposed models and the *pipeline model*, any potential gain for these scenarios is best addressed by the company using techniques to increase instance size (such as reducing the frequency in which the problem is run). The proposals to address improvements for small instances are not part of the scope of this thesis.

During the operation, large instances occur almost exclusively during rush hours, in which the volume of *assignables* may temporarily surpass by a considerable amount the number of available *shoppers*. When the ratio of *assignables* to *shoppers* is greater than 1.0 then there are *assignables* that will inevitably be left un-assigned until an instance where there are enough *shoppers* available.

For the results shown in this thesis the point of reference is considered as the number of *assignables* that require a match in either an instance or run instead of the theoretical limit for the potential match count (e.g., if an instance has 10 *assignables* and 4 *shoppers* and 3 matches are found, the reported allocations is 30%, even though 75% of the apparent maximum of matches was achieved).

The reason for reporting results using *assignable* count as a baseline is to have results be comparable across models. The theoretical allocation count depends on the exact sequence in which both *assignables* and *shoppers* were available. Even though the starting point and base data may be the same across models, the allocations or lack thereof made by a model could make the maximum *assignable* count that could be allocated different to another of the evaluated models.

The results shown are the aggregate for the scope as previously mentioned, the matches for a particular *assignable* may happen in different instances for different models, meaning that an *assignable* may be matched on the n^{th} instance with one model and on the $n + i$ for another model.

Since the results are shown regardless of which was the instance of when the *assignable* was matched, any dynamic programming effects (if the instance contained efficient or mostly inefficient match candidates) are captured by the efficiency difference between models, this allows the results to show if a model tends to generate convenient scenarios or not.

5.1.5 RESULTS FORMAT

The metrics for each result are shown with each metric's average value in the corresponding table for each run. The mean of each metric only takes into account the values for *assignables* with an assigned *shopper* unless explicitly stated otherwise (for the *all allocation tries* of chapters 5.4 and 5.5 metric).

The results plot shows the cumulative allocation distribution for the plotted metric. For any given model, the plot shows in the y axis what percentage of the total available *assignables* that were satisfied for a given metric value on the horizontal axis.

The larger the allocation coverage for a given value of the metric, then the model is considered to find allocation at a lower allocation cost than one with a lower coverage. Lower allocation coverage implies that the model achieved a lower total coverage or that the allocations found were found at a larger cost than the model with the larger allocation coverage.

For this thesis, the metric of *total time cost*, which is the company's definition of allocation cost, is show within the body of the document. The result for *allocation tries*, as well as the breakdown of *total time cost* per components is shown in section 8.3 of the document annex.

5.2 MEASUREMENT METRICS

The model used by Cornershop does not utilize an objective function in the same way as it is applied in a mathematical optimization problem, this presents a challenge for evaluating the quality of the solutions provided for each model.

For the mathematical optimization model, utilizing the cost of the objective function (since it is a *min cost* O.F. formulation) is also deceiving. For a poor ratio of *assignables/shoppers* a sizable portion of the objective function total cost could be explained by the opportunity cost component for non-allocated *assignables*, which is a new concept introduced for the mathematical formulation proposals.

The cost of the objective function can be made arbitrarily large for instances with *assignables* left un-assigned, since the opportunity cost could also be defined as an arbitrarily large value.

The metric used for selecting the most efficient allocation cost, which for this thesis is referred to as ***total time.*** The *total time* cost is not directly translatable to operational costs for the company, as it includes the business skews in the form of handicaps as seen in equation (7) in chapter 3.2.4.

The definition of *total time cost* can be seen in equation (8) in chapter 3.2.4 and is counted in minutes for units.

It is not the purpose of this thesis to determine whether the expected outcome of the *handicap* component of the *total time cost* correctly captures business preferences, as this is the way the problem is modeled for the *pipeline model*.

5.3 SINGLE INSTANCE RUN

The instance results don't deal with any communication across instances, therefor the problem solved by all models is the same, in that the same match candidates are used as input. The only differences in allocation options are for the *pipeline model,* which may discard feasible match candidates when applying filters that are geared towards improving code running time, as the company implementation does.

The greedy characteristics of the *pipeline* model could also yield a scenario where the feasible *shopper-branch* allocation candidates for an *assignable* were assigned to another, leaving the *assignable* without a feasible match for that instance.

5.3.1 RESULTS

Zone 1 instance results

Zone 1 is one with a large business volume: the mathematical optimization formulations for this zone have approximately 500 constraints and 8000 binary variables.

As seen on table 1, the solution of the *pipeline model* provides considerably more efficient matches on average than the proposed optimization model. The *pipeline model* achieves this efficiency primarily by allocating shoppers that are closer to the assigned store and have a smaller handicap

cost, which would indicate that these matches also represent ones that are more in line with business preferences.

The greater average efficiency of total time cost for the *pipeline model* comes at the expense of the estimated travel time from each store to the customer delivery location, indicating a preference for stores that are, on average, further away from the delivery location.

Table 1 | Zone 1 – Instance average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	5.618	5.618	5.618	5.618	8.125
b2c time	12.620	12.620	12.620	12.620	14.539
handicaps	12.542	12.542	12.542	12.542	5.736
Total time	30.780	30.780	30.780	30.780	28.401
Allocation coverage	48%	48%	48%	48%	13%

What is seemingly a more effective use of resources, due to a lower mean *total time* cost, does not consider the allocation coverage that each model achieves for the instance. The *pipeline model* manages to match a lower percentage of the instance *assignables* for almost any value of the total time cost as seen on Figure 1.

In essence, even though both approaches are solving the exact same instance, the behavior of the mathematical optimization models could be best described as a *Maximize the number of participants,* since it yields a match for a much greater percentage of the instance *assignables* with worse average total time cost.

The *pipeline model* behavior is best described as a *Minimize system wide travel time* when un-matched *assignables* are not considered within the total cost, since it achieves a lower average cost considerably lower match coverage.

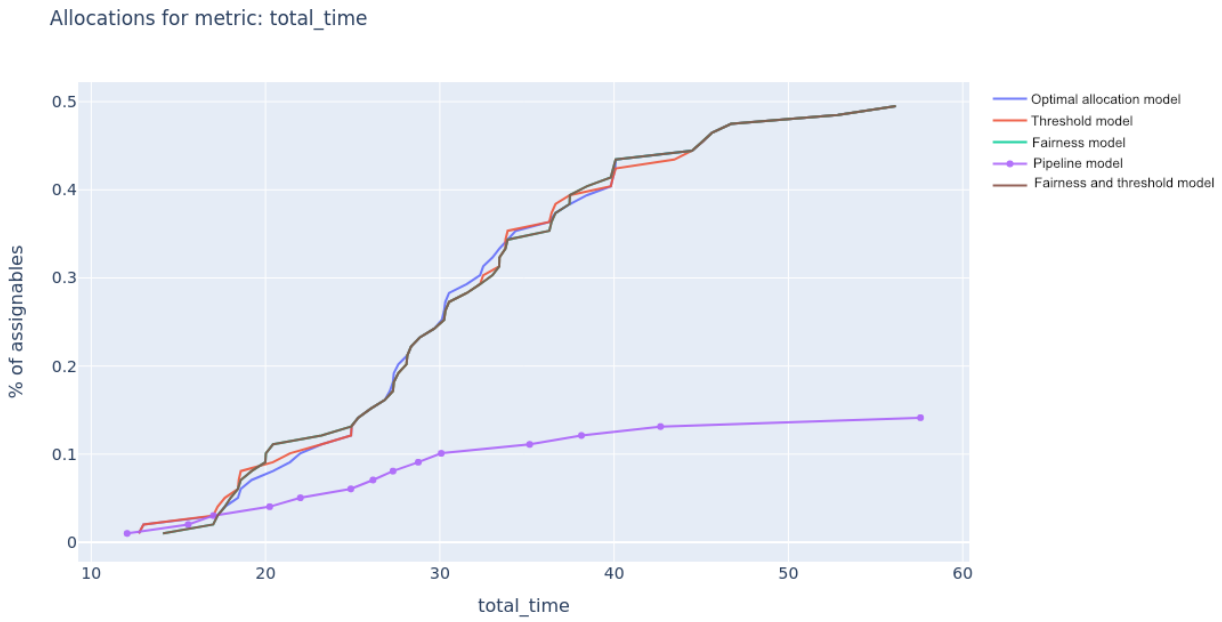
The tradeoff between coverage and average **total time cost** performance is expressed as patience in the *pipeline model* to defer the allocation of an *assignable* for a later instance. This could lead to a more efficient aggregate result if the deferral allows for more convenient future instances, although there is no guarantee of this being the case. This “patience phenomenon” could also result in drawbacks of orders arriving late or with higher costs if future instances have mostly “expensive” options to match the *assignable*.

The cumulative allocation cost (total time cost) for the *assingables* of the shown instance, shows the percentage of *assignables* where a match was found for a given value of total time cost. The curve is composed of the sorted cost for each *assignable* in the instance where a match was found.

It is worth pointing out that as previously mentioned, the distribution is shown for all *assignables* in the instance, whilst there could be less *shoppers* than what is required to match all *assignables* in the instance.

For the case of Zone 1 during a noon instance, approximately half of the instance *assignables* had a shopper that could have completed the job, therefore the distribution curves stop below the 100% mark.

Figure 1 | Zone 1 – Noon instance cumulative distribution of total time cost



Note: results for both heuristics with "fair" criterion have overlapping performance curves

For the performance of Zone 1 during noon, all mathematical optimization derived models perform similarly, however there is a significant difference between these and the *pipeline model*, which allocates *shoppers* with an increasing marginal cost, meaning that each additional *shopper* it is "willing" to allocate is considerably more "expensive" than the previous, while the proposed formulations have a more gradual increase in cost for each additional matched shopper.

The results shown for zone 2 have a median of approximately 610 variables and 480 constraints and a max of approximately 2200 variables and 640 constraints.

Zone 2

Figure 2 | Zone 2 - Noon instance cumulative distribution of total time cost

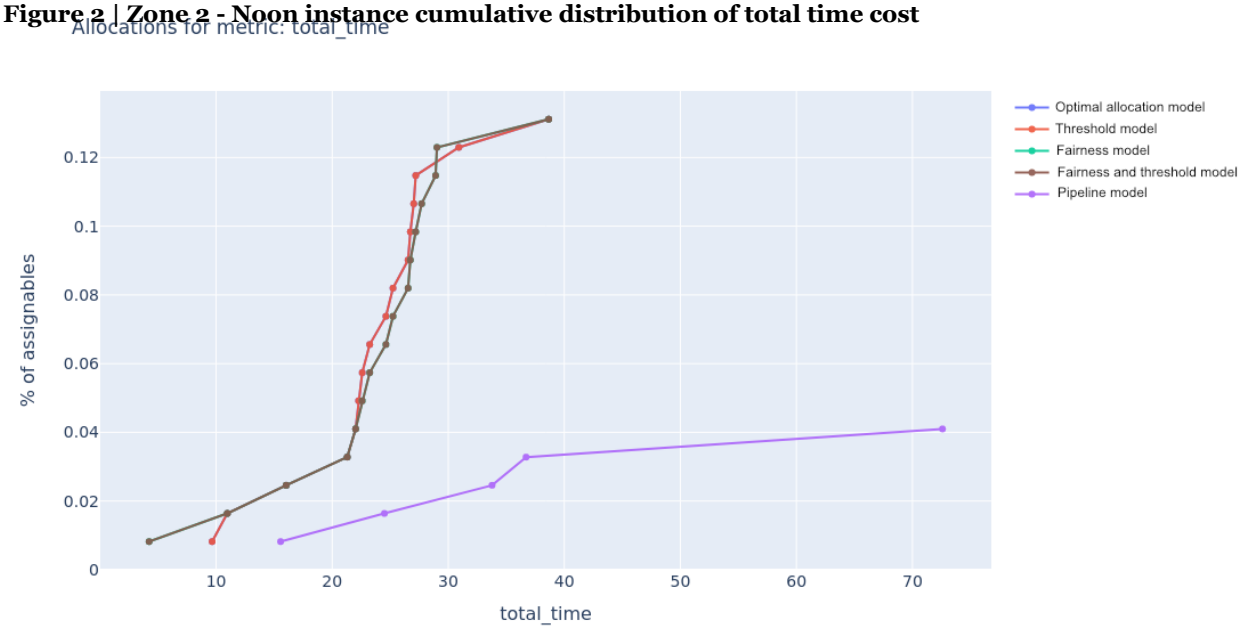


Table 2 | Zone 2 – Instance average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	9.432	9.432	9.432	9.432	8.908
b2c time	9.160	9.160	9.160	9.160	9.237
handicaps	4.847	4.847	4.847	4.847	18.482
Total time	23.439	23.439	23.439	23.439	36.626
Allocation coverage	13%	13%	13%	13%	5%

From the performance seen in Figure 2, the mathematical optimization derived models perform significantly better that the current greedy approached used by the company, however unlike the performance curve in Figure 1, the marginal cost for allocations follows a segmented growth pattern, where incremental allocations may have very different marginal costs.

For the case of this instance, the optimization models find additional matches at a very small incremental cost between the values of 20 and 30 total time minutes. The model used by Cornershop requires accepting very high allocations costs to improve the coverage of found matches past the value of 40 *total time* units.

The reason for this stepped behavior for performance curved is explained by the non-normal distribution of shoppers within a zone, which in operational terms is expressed as small clusters of shoppers that usually are positioned around high job volume locations.

The extent to which these clusters affect matching performance in a "shopper starved" instance as shown in Figure 2, depends on the predominance of orders that are fulfilled at stores with more clustered shoppers for which the model manages to find a match for.

Zone 3

This "resource" imbalance for a particular instance is common in zones that have a larger sales volume and have concentrated demand peaks. The uneven distribution of *assignables* amongst select instances can be attributed to the way in which the company offers promised delivery times.

Customers that buy using programmed[12] orders are offered the same promise windows, which results in clusters of similar orders requiring a match within a short time span. The company has measures to alleviate this phenomenon, but the effect will naturally be present, even though it's attenuated.

The instance of zone 3 has approximately 500 constraints and 1000 binary decision variables.

Unlike the results for zones 1 and 2 which are scenarios with an unfavorable *assignable* to *shopper* ratio, the result for zone 3 has a favorable ratio (there are enough *shoppers* to match most or all *assignables*), which translates into a shift of model behavior of "which orders are most efficient to have allocated" to "how can be the orders most efficiently allocated".

From a model allocation capability standpoint, this is a more interesting result, since it leverages the combinatorial nature of the assignment problem to a greater extent than the first to two runs.

Intuitively, for a single instance a higher number of matches found should normally be interpreted as a better result, however from a "global efficiency" standpoint that is not necessarily the case, since neither of the current and proposed formulations make use of the online nature of the recurring assignments across instances.

It's not too hard to realize that a less match coverage could end up "being a feature, not a bug", since this could result in "smother" scenarios faced on a sequence of instances, where many

[12] The company service offers to place an order as a scheduled order or on-demand, which for practical purposes is equivalent to a "within the next X minutes" promise. For matching, the only difference between each type is in the value of handicaps for the assignable.

allocations during instance *n*, may result in only having inefficient combinations be a more common occurrence since more of the earlier orders were allocated in a single instance.

Zone 3

Figure 3 | Zone 3 - Noon instance cumulative distribution of total time cost

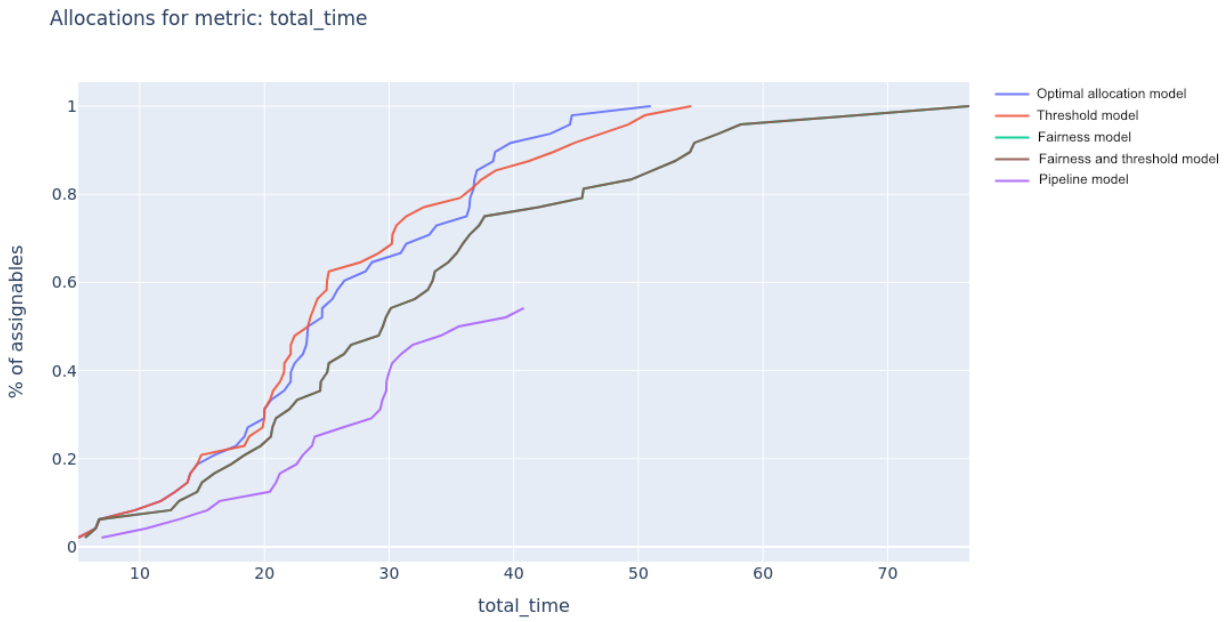


Table 3 | Zone 3 – Noon - Instance average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	4.663	4.636	10.338	10.336	2.925
b2c time	14.670	14.670	14.670	14.670	15.800
handicaps	6.531	6.531	6.799	6.799	6.835
Total time	25.837	25.837	31.808	31.808	25.561
Allocation coverage	100%	100%	100%	100%	58%

5.4 RUN RESULTS

5.4.1 SEQUENCE DATA

As described in chapter 4.2, a sequence of instances is generated using historical scenarios derived from the company's operation. Like previously mentioned, resources allocated during an instance are removed from all following instances regardless of them appearing in future logs or not, likewise resources left idle or un-allocated on instance are then added to following ones.

This process of instance inheritance goes on until the resource in question is allocated to another or the end of the run.

For **allocation tries**, which again represents the number of times an *assignable* was present in an instance (or the number of times a match was attempted), there are two metrics:

- **Assigned Allocation tries**: the match tries count for *assignables* where a match was found within the sequence
- **All allocation tries**: the match tries considering all *assignables* regardless of if a match was found within the sequence or not (this considers un-matched *assignables*)

It is important to remember that as with the single instance results, a sequence may not contain enough shoppers to allocate all *assignables* (*shoppers* can only be allocated a single *assignable*), however the results are presented with regards to the assignable count, which may be larger than the maximum feasible allocation count.

5.4.2 RESULTS

The results presented show 3 types of scenarios:

- A non-strained mature market
- A strained mature market
- A new market that can be considered not to be shopper strained

5.4.2.1 Non-strained mature market

A mature market has a larger order volume than a typical new market. Mature markets have had an active operation for longer, this affects the location of *shoppers* thought the city as well as having more of them on average.

For mature markets more experienced *shoppers* will have a tendency to position themselves close to large order stores, as well as having a larger proportion of experienced *shoppers* in comparison with newer markets.

The results shown for a non-strained operation exemplify a scenario where for most instances there are enough available *shoppers* to fulfil most if not all the *assignables* within the instance.

The instances shown for a non-strained mature market have median of approximately 100 constraints and 260 variables and reach a maximum of approximately 500 constraints with 1000 variables.

5.4.2.2 Strained mature market

Like the characteristics of a non-strained mature market, a strained shared the *shopper* positioning and seniority characteristics mentioned in 5.4.2.1.

Where the scenarios differ is in the ratio between *assignables* and *shoppers* as well as the size of instances. The results show typically have less *shoppers* than needed for most instances, which results in larger allocation tries count[13].

Un-matched *assingables* are usually matched at later instances past the daily rush, which due to the length of the shown runs is not reflected on the results. For this zone the elapsed time between an *assignable* being available for a match to when it's matched is particularly relevant since this negatively affects if an order will arrive late or not. From the results shown the allocation tries directly correlates to longer match wait times.

The instances for this zone have a median of approximately 500 constraints and 2500 variables and a maximum of approximately 650 constraints and 13000 variables.

5.4.2.3 New market

Contrary to the behavior of mature markets, where the operation has a subset of partners with large stores that concentrate the demand fulfilling large orders, new markets operate with a larger participation of smaller stores. For matching purposes, it implies that there is less of a clear pattern of experienced shoppers around a small subset of store branches.

Unlike mature markets, the "recently" opened ones have less volume of both *shoppers* and *assignables*, which leads to smaller instance sizes. For the results shown, the median instance has approximately 100 constraints and 10 variables with a maximum of approximately 200 constraints and 450 variables.

These three scenarios are shown in a rush hour and normal operation setting (which are represented by a morning and afternoon run respectively). These configurations are representative of the main "use cases" that are relevant for the operation.

Mature non saturated market

Zone 3 is a large and "mature" zone, with a large volume of both *assignables* and *shoppers*. It is considered a zone that normally is not "shopper strained". For the context of this thesis the main difference between mature markets and new operations is in the volume of orders that are allocated within any given date, where the mature market handles a considerably larger volume of both *assignables* and shoppers.

Although not formally studied within the scope of the thesis, mature markets also have more experienced shoppers, which for matching purposes is manifests in shoppers accumulating around stores that are perceived to yield a better compensation (this is explained by stores with larger orders and a higher chance of a multi order assignable from some stores). From a results

[13] Allocation tries refers to the number of instances an *assignable* goes through before being matched.

point of view this phenomenon is seen as a more stepped performance curve for *shopper to branch* travel times.

Zone 3

Figure 4 | Zone 3 - Morning - Run cumulative distribution of total time cost

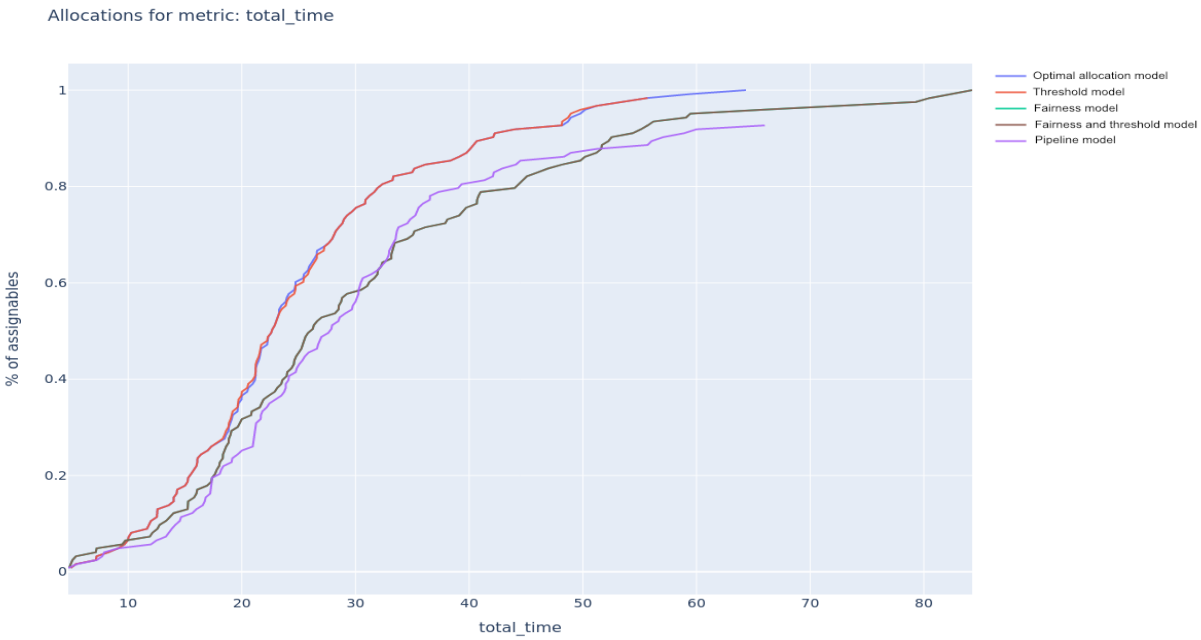


Table 4 | Zone 3 - Morning – Run average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	4.506	4.325	9.764	9.764	5.075
b2c time	13.700	13.325	13.692	13.692	13.570
handicaps	6.860	6.788	7.197	7.197	9.094
Total time	25.065	24.439	30.653	30.653	27.739
Assigned Allocation tries	1.000	1.000	1.000	1.000	1.439
All allocation tries	1.000	1.130	1.000	1.000	1.797
Allocation coverage	100%	98%	100%	100%	91%

The overall performance of the mathematical models that don't add a *fair heuristic* show a steeper performance curve, particularly the *threshold model* manages to allocate all *assignables* in the sequence. The more efficient allocations for the "pure optimization" models also manage to achieve greater coverage.

As discussed within the single run results, this sequence results clearly shows that any "resource smoothing" benefits derived from allocations with less coverage are not enough to compensate for the per instance gains of the mathematical formulations.

Figure 5 | Zone 3 – Noon – Run cumulative distribution of total time cost

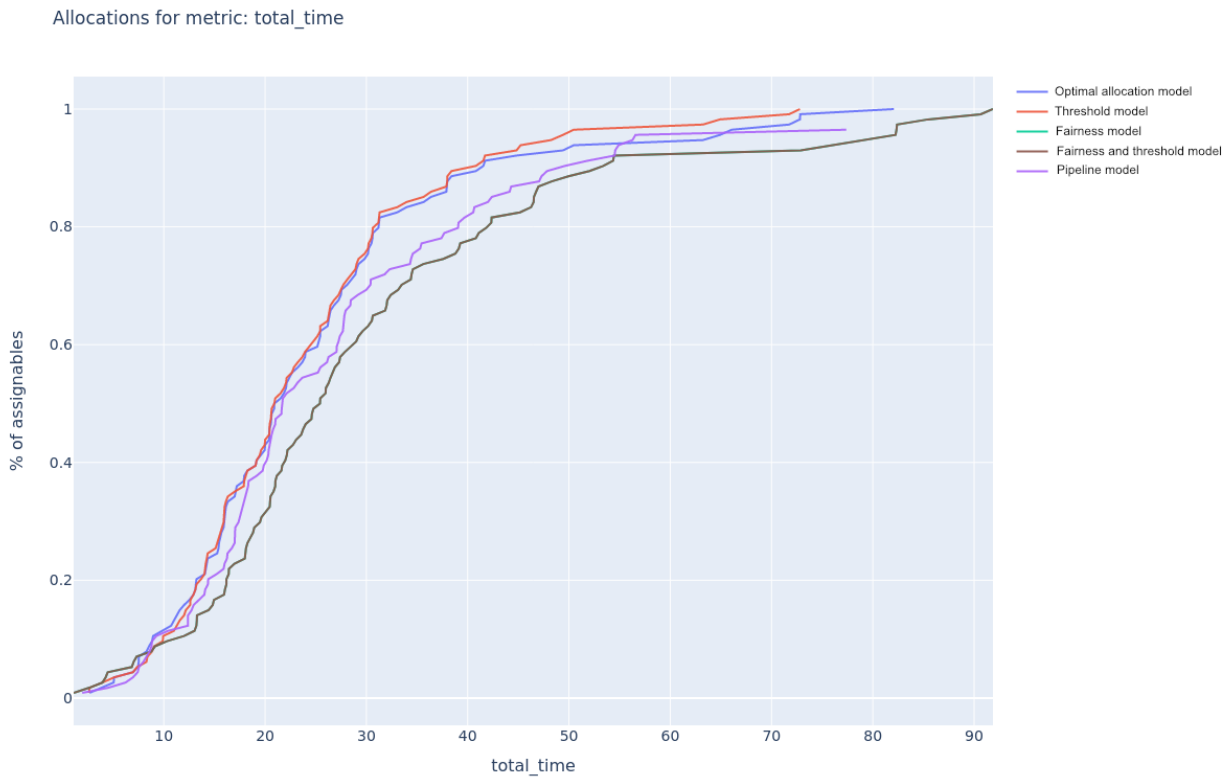


Table 5 | Zone 3 – Noon – Run average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	4.003	3.700	9.383	9.383	3.790
b2c time	15.075	14.455	15.237	15.237	14.172
handicaps	5.723	5.741	5.672	5.672	7.393
Total time	24.800	23.896	30.291	30.291	25.356
Assigned Allocation tries	1.105	1.544	1.105	1.105	1.509
All allocation tries	1.105	1.544	1.105	1.105	1.737
Allocation coverage	100%	100%	100%	100%	96%

For morning results of zone 3, the *total time* gains are primarily explained by a reduction in *shopper to branch* times and *handicaps*. The average *branch to customer* time is practically unchanged, this means that the gains are explained by a more efficient allocation of shoppers both in placement and characteristics, rather than fulfilling the orders at a closer store or with different transportation method than those assigned by the *pipeline model*.

The results for the afternoon run shows less improvement than the rush hour morning run, however the *threshold model* does manage to extract additional efficiency from the allocations, although in doing so it is also the only mathematical optimization approach with lingering orders that take longer to assign, particularly in the group of orders where a match is not found within the run.

The advantages for the afternoon run are mainly explained by a lower average handicap cost, representing allocations that on average are better aligned with business preferences in comparison to the pipeline model.

The average *shopper to branch cost* is higher than the company heuristic for all proposed formulations except the *threshold model*, the loss in shopper to branch time is especially relevant for the fair selection heuristics, suggesting that the proposed heuristic does not adequately encapsulate the "fairness" criterion to matches that are "close" to the efficiency optimized allocations as is the case for the company heuristic.

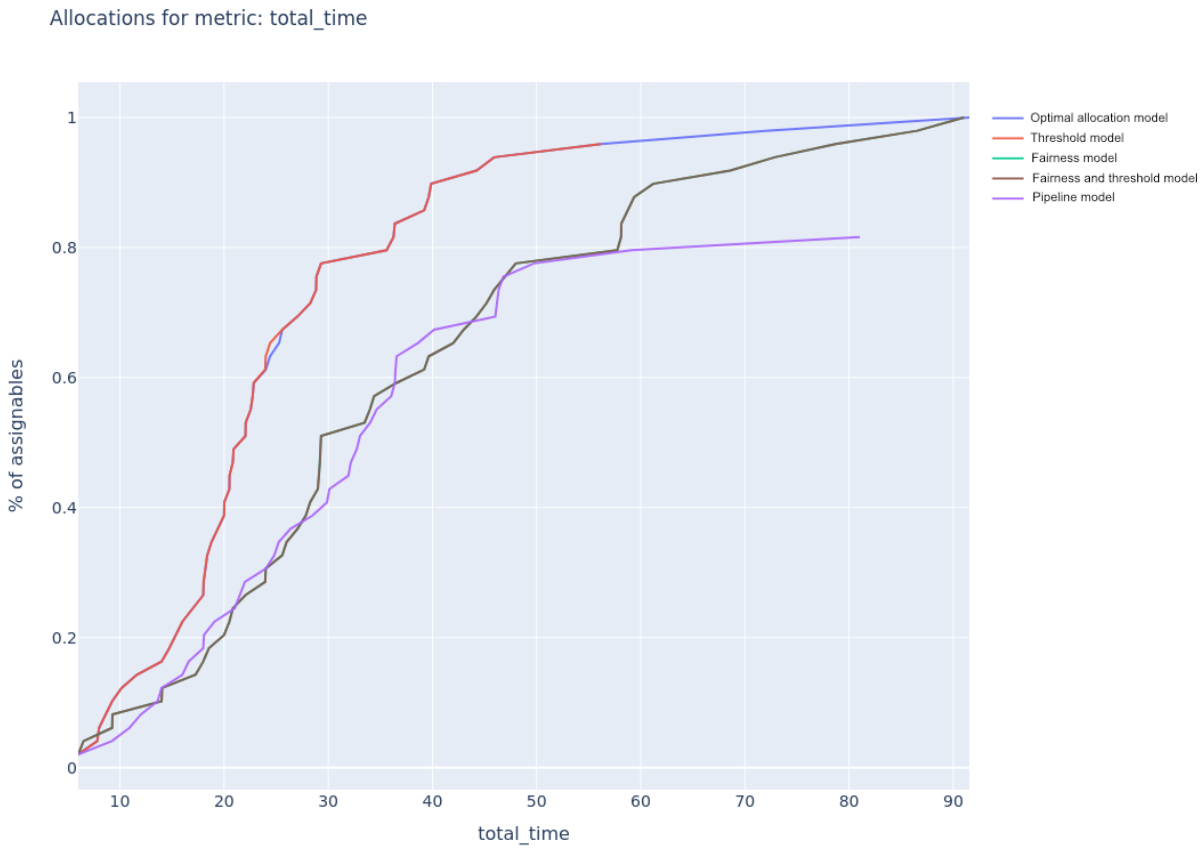
New market

Zone 4 is an example of the newer markets[14], this example is a larger case amongst new markets, however the volume of both assignable and shoppers for these instances still lag behind that of more mature markets.

Unlike mature markets, *shoppers* for newer markets are expected to not be as accustomed to working with the platform. The unfamiliarity results in the distribution of the shopper locations being less clustered around larger stores.

Many of the newer markets where the company operates also have relevant competitors with whom shoppers may have experience, how this past shopper experience affects matches is not formally studied.

Figure 6 | Zone 4 - Morning - Run cumulative distribution of total time cost



[14] Most of the cities where cornershop operates where added during the last 2 of the 7 years of operation, meaning that these zones are from cities where the operation started during the pandemic.

Table 6 | Zone 4 - Morning - Run average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and Threshold model	Pipeline model
s2b time	6.239	4.767	17.003	17.003	11.673
b2c time	14.944	14.018	15.013	15.013	14.166
handicaps	4.659	4.678	5.106	5.106	4.549
Total time	25.842	23.463	37.121	37.121	30.388
Assigned Allocation tries	1.000	1.000	1.000	1.000	1.225
All allocation tries	1.000	3.041	1.000	1.000	3.694
Allocation coverage	100%	97%	100%	100%	83%

The performance curve for zone 4, as with the performance for Zone 3, has a steeper curve and better coverage for the mathematical optimization-based approaches versus the pipeline model.

Unlike a similar performance similarity between the *fairness selection* and *pipeline models* seen for zone 3, this zone achieves greater coverage with the optimization approach for all models.

It is worth noting that the average total time cost for *threshold model* on the morning run, is up to 22% lower than that achieved by the *pipeline model*. The *optimal allocation model* is on average more than 14% more efficient in terms of total time cost than the *pipeline model,* while having both better coverage and less *allocation tries* for both allocated and non-allocated *assignables*.

Figure 7 | Zone 4 – Noon – Run cumulative distribution of total time cost

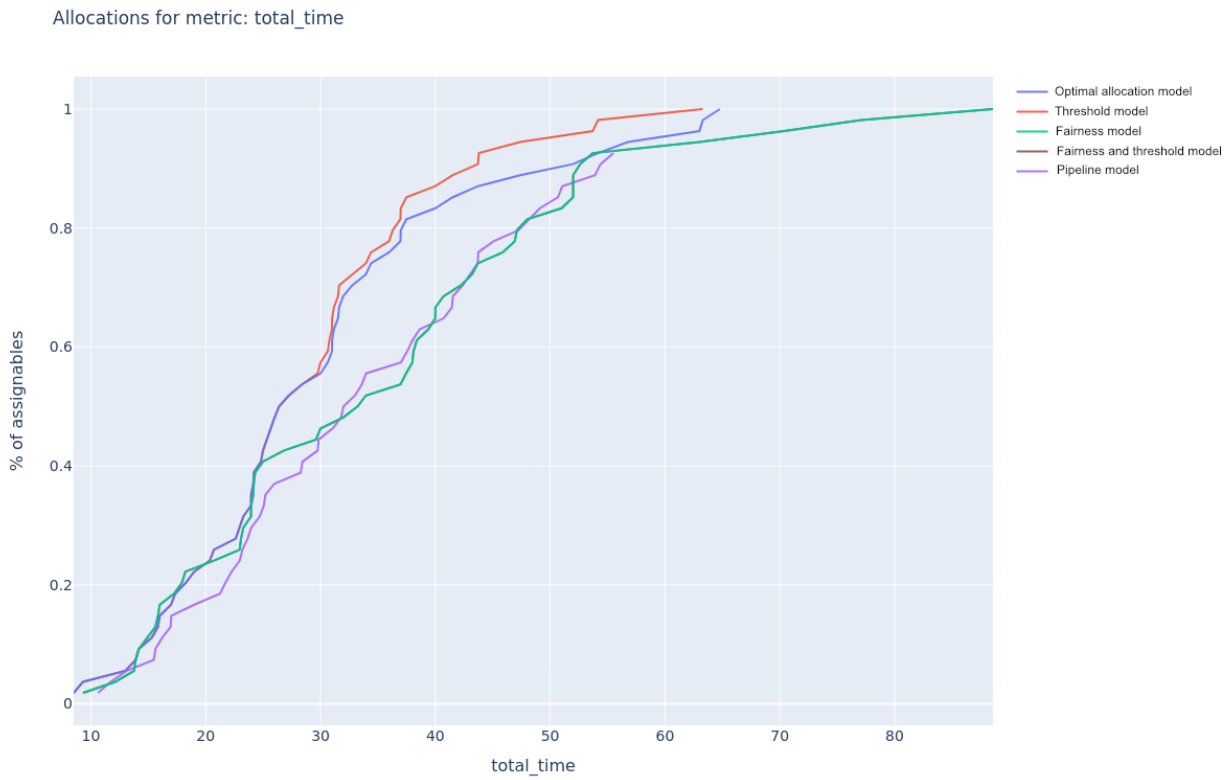


Table 7 | Zone 4 – Noon – Run average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	12.227	10.814	16.174	16.174	14.258
b2c time	13.042	13.042	13.173	13.173	12.615
handicaps	5.107	4.900	5.355	5.355	4.749
Total time	30.376	28.756	34.702	34.702	31.623
Assigned Allocation tries	1.000	1.204	1.000	1.000	1.240
All allocation tries	1.000	1.204	1.000	1.000	1.537
Allocation coverage	100%	100%	100%	100%	92%

Shopper strained mature market

Zone 1 is amongst the largest subdivisions the company has in its operation, it is a mature market and from a matching point of view shares, many of the characteristics as Zone 3. Unlike Zone 3, the balance of *assignables* and *shoppers* is skewed heavily towards more *assignables* for the peak demand slices of the daily operation.

Figure 8 | Zone 1 – Morning – Run cumulative distribution of total time cost

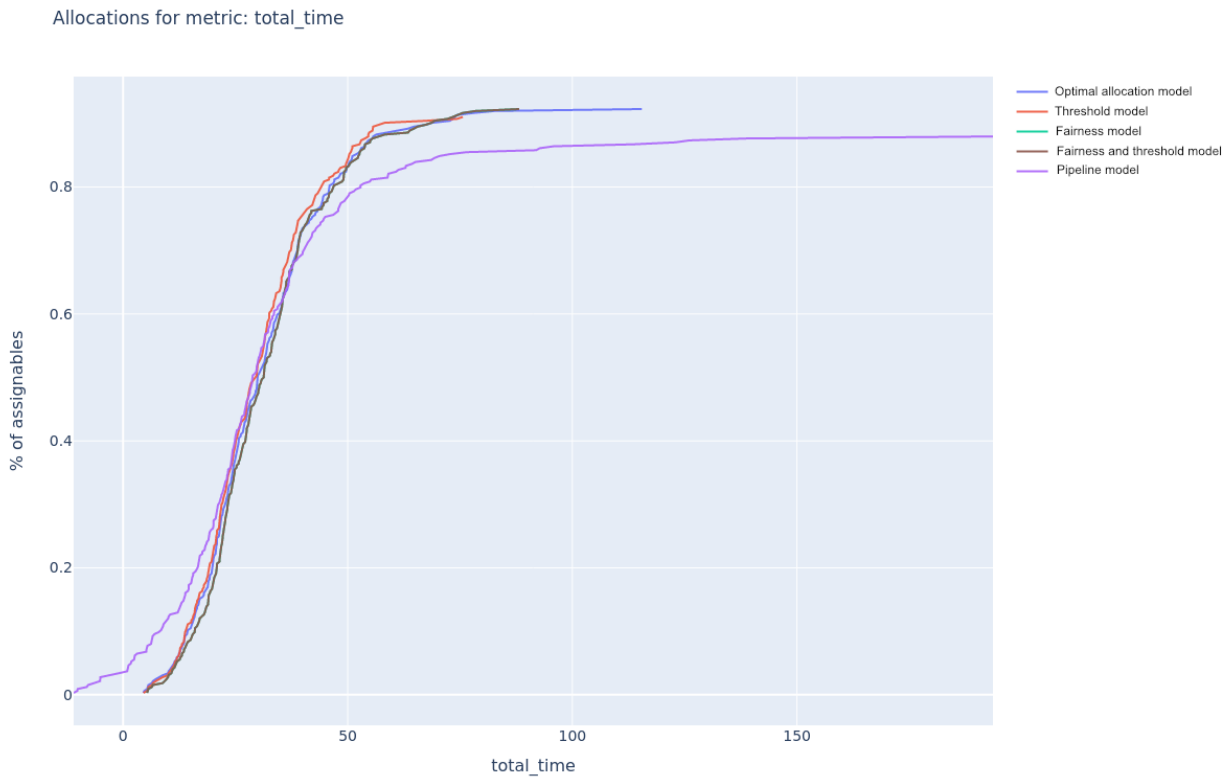


Table 8 | Zone 1 – Morning – Run average performance component summary

	Optimal allocation model	Threshold model	Fairness model	Fairness and threshold model	Pipeline model
s2b time	7.508	6.547	8.236	8.236	4.458
b2c time	13.135	12.360	13.038	13.038	28.161
handicaps	10.191	10.077	10.170	10.170	-2.889
Total time	30.833	28.984	31.444	31.444	29.731
Assigned Allocation tries	1.261	1.417	1.284	1.284	3.484
All allocation tries	2.074	2.346	2.096	2.096	4.457
Allocation coverage	93%	89%	94%	94%	85%

The performance for the morning is the only zone shown where the performance curve for the *pipeline model* outperforms all the mathematical optimization approaches under at least a portion of the allocation costs.

For the morning run shown, the performance of optimization models achieves better coverage than the *pipeline model* only above approximately 30 minutes of *total time* cost, which roughly coincides with the average performance of the models shown. This performance means that for allocations above the mean the optimization models manage to find more matches whilst the *pipeline model* has its matches skewed towards values below the mean.

The performance of all but the *50 threshold model* is on average, worse than the company heuristic. This difference is primarily by a much better average performance on the *handicap* component of the *total time* costs, meaning that these allocations are better aligned with business preferences, whilst the optimization models show lower cost for *branch to customer* delivery times.

The swap of better *handicap* cost for longer trips can be explained by a "feature" of the *pipeline model* not added to the global optimization approach, where there is an additional large handicap added for the merchant that is associated with the shopper app of the customer who placed the order.

The more efficient allocations of the *pipeline model* are not a free lunch, the cost benefit comes at the expense of considerably more allocation tries, which directly translates to longer order matching times even for the worst-case scenario of the optimization models. An allocation is attempted on average twice as often for the *pipeline model* as for the proposed mathematical optimization versions.

The longer allocation times for orders are related to a higher chance of the orders within the *assignable* arriving after the promised delivery window. Although the delivery windows are not directly considered in the objective function of any of the models (adding these as a cost should be high in the priorities for future work), the fulfillment of the delivery window is the main "order quality" metric for the operation alongside order fill rate[15].

[15] Fill rate: the percentage of items within an order where the order items were found, or a suitable replacement item was accepted by the customer.

Figure 9 | Zone 1 – Noon – Run cumulative distribution of total time cost

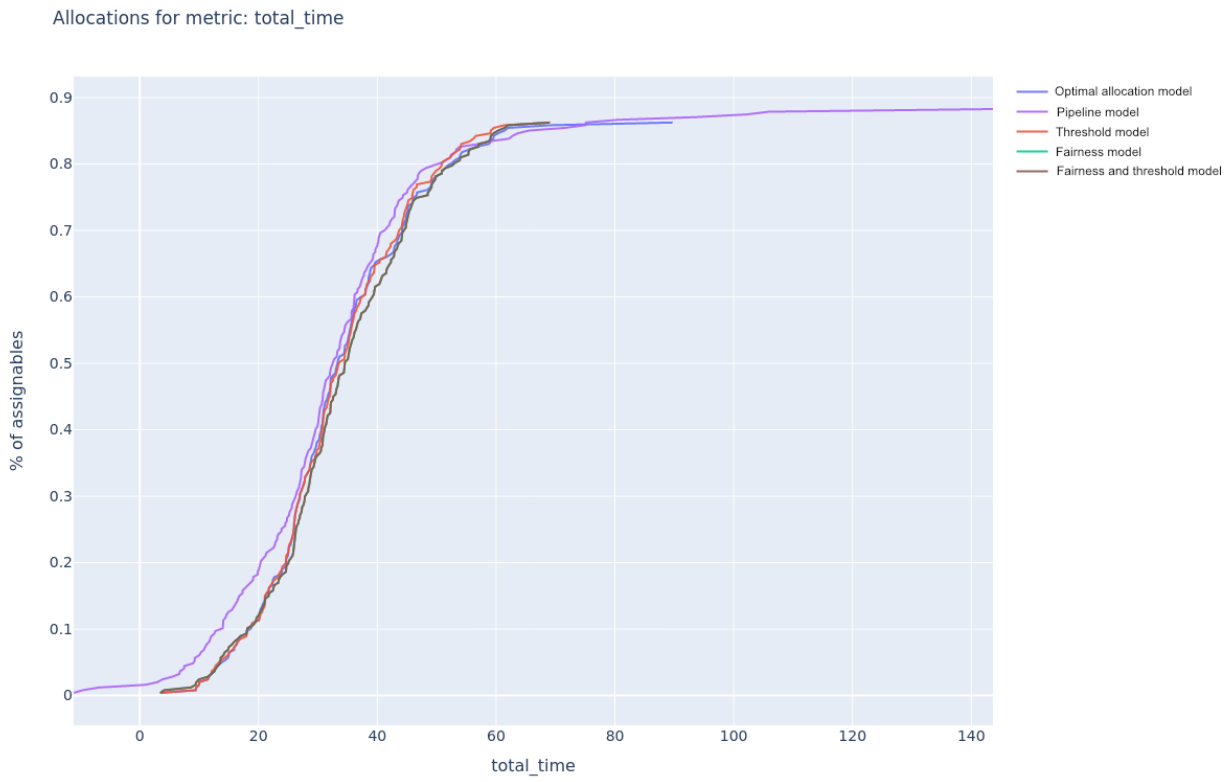


Table 9 | Zone 1 – Noon – Run average performance component summary

	Optimal allocation model	**Threshold model**	**Fairness model**	**Fairness and Threshold model**	**Pipeline model**
s2b time	8.021	7.847	8.625	8.625	5.216
b2c time	14.198	14.198	14.178	14.178	25.316
handicaps	10.555	10.418	10.452	10.452	1.603
Total time	32.774	32.462	33.256	33.256	32.135
Assigned Allocation tries	1.338	1.390	1.343	1.343	2.179
All allocation tries	2.251	2.296	2.255	2.255	3.453
Allocation coverage	87%	85%	87%	87%	87%

The afternoon run for Zone 1 represents a run that although large, is considerably smaller than the morning scenario for the same zone. The results for the *pipeline model* for this run are the only results shown where the *pipeline model* achieves both more efficient allocations average allocations and greater coverage than that of all the proposed models.

In a similar fashion as the results shown for the morning run, the greater efficiency of the *pipeline model* comes at the expense of longer "matching times", seen in more *assigned allocation tries*. The *assigned allocation tries* difference between models is smaller than for the morning run, where the worst case is on average 1.6 times the count of that of any other result for this run.

The aggregate result for *assigned allocation tries* is approximately 1.5 times the average result of the aggregate tries for the optimization models, which is also a smaller difference than the morning run result at 2.6 times.

The greater efficiency for the *pipeline model* can attributed to a much smaller result for the *handicaps* component of the allocation cost, at over 6 times the component cost for the mathematical optimization models.

The transportation costs for the mathematical optimization models are, as with the morning sequence, smaller than the value attained by the *pipeline model*, where the heuristic approach component cost is approximately 70% larger than that of the mathematical optimization values.

It is worth nothing that the performance curve of the afternoon run shows a better result for the last percentage of allocations, where the *pipeline model* requires finding matches at over twice the *total time* cost than that of the mathematical optimization proposals. This behavior is very much in line with expected what is expected for a greedy approach Vs. that of ones which at least on a per instance level find the global optimum matches.

5.5 CITY MODEL

As described in chapter 5.1.1, the company does not solve the shopper allocation problem with the same scope as the customers might perceive service availability. The actual scope of each instance is defined by a polygon that may cover the entire availability zone for Cornershop' s service or a partition of the availability zone.

For illustration purposes, since they don't show real zones, figure 10 illustrates the availability of the company's service for Santiago Chile, whilst figure 11 illustrates allocation zones.

Figure 10 | Whole city example zone scope for Santiago, Chile

Figure 11 | City zones example scope for Santiago, Chile

Although running the shopper allocation problem with a different scope as is used by the company is not a strictly speaking an algorithm improvement, enabling the application of business filters discussed in 3.1 in parallel would allow the implementation of this larger scope within company compute time benchmarks, whilst the *pipeline model* can't reliably perform with the larger scope.

The subdivision of an operating zone into smaller chunks can be seen as equivalent to solving the entire operating zone (the entire city), with the added constraint that all resources must intersect with a smaller polygon, where the smaller polygons are the current operating zones in cities where the operating zone does not cover the entire city.

On a purely mathematical point of view, it can be trivially proven that each instance will have a lesser or equal cost on the whole city formulation in comparison to the sum of the cost for each individual partition, as the optimal solution of the subdivided problem is also a feasible solution of the city problem, but the contrary is not necessarily true[16].

This thesis focuses on the implications of switching algorithm for the company's application and not necessarily the solution of each instance. As seen for the results of a strained mature market in 5.4 an algorithm that achieves a per instance improvement will not necessarily yield less costly results for a run of consecutive instances, although this is expected to be the outcome.

Operating zone resources are defined without overlap, meaning that the instances of each operating zone are completely independent. For the results of running a city model the **whole city model** is defined as the problem without the zone subdivision, so the operating area for this problem covers the entire city.

The **regular model** is defined as the sum of outcomes for all the zones in the same city as for the city mode. The remaining configuration and presentation of results is kept the same as for the runs results (chapter 5.4).

Due to the city model being a comparison without an algorithm change, only the results for the base mathematical optimization algorithms will be shown.

[16] Instances where the feasible matches are the same set for the divided and whole problem are equal can be built, however there is no guarantees of this being the case for an operational scenario

Figure 12 | Zone 4 – Run cumulative distribution of total time cost

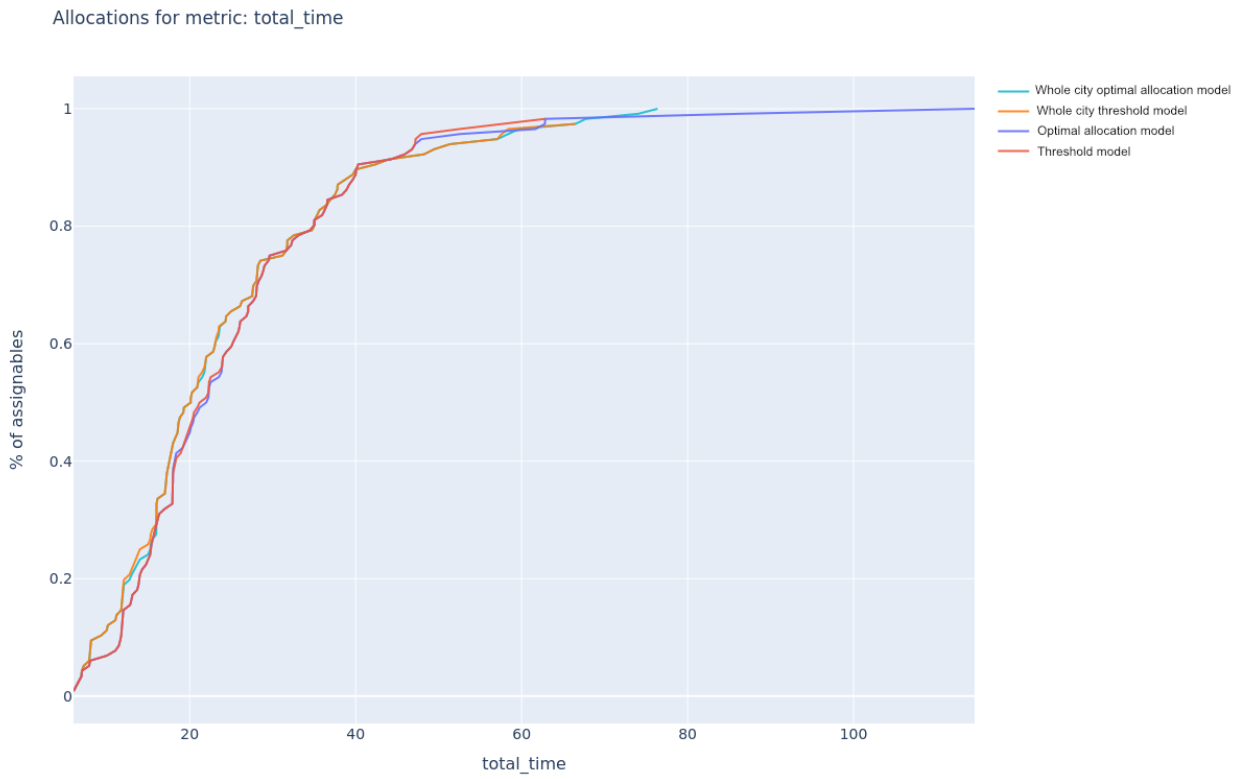


Table 10 | City run average performance component summary

	Optimal allocation model	Whole city optimal allocation model	Threshold model	Whole city threshold model
s2b time	6.512	4.765	5.557	4.174
b2c time	14.242	14.315	13.781	13.795
handicaps	4.658	5.245	4.556	4.988
Total time	25.412	24.325	23.895	22.956
Assigned Allocation tries	1.216	1.000	1.360	1.372
All allocation tries	1.216	1.000	1.750	1.862

5.6 RESULTS ANALYSIS

5.6.1 INSTANCE RUNS

From the results seen in 5.3, assessing the performance of a model for the *shopper* allocation problem is not as straightforward as selecting the one which achieves the lowest average allocation cost, as other aspects such as allocation coverage and allocation time must also be taken into account.

The *pipeline model* for zone 1, finds allocations that on average, have lower *total time cost* than the proposed formulations, however the match coverage for the instance is considerably smaller than for any of the proposed models.

It is not evident if the result achieved with smaller total time cost and less coverage has a net positive or negative effect on outcome for a sequence of instances. The deferral of allocating less efficient allocation candidates could allow for less costly allocation candidates to become available in a future instance, as well as setting the future instance with comparatively less convenient allocation candidates.

For instances where there are enough *shoppers* to fulfill either the majority or all the instance *assingables*, the mathematical optimization models outperform the *pipeline model* as seen with the results for zone 2 of chapter 5.3.

The performance of chapter 5.3 suggests the mathematical optimization proposals perform better when the instance allows for the majority of *assignables* to be fulfilled. The result from single instance can be explained by the model "sacrificing" the *total time cost* of *assignables* to avoid having to the non-allocation cost for *assignables*.

The described behavior suggests that future work geared towards a more informed management of the non-allocation decision could yield important average matching cost benefits.

From the results of single instance runs it can be concluded that the performance of mathematical optimization models is mostly positive in comparison to the *pipeline model*, however the operational efficiency in the matching cost (*total time cost*) for an average *assignable* depends on both the *assignable/shopper* ratio and if the smaller match coverage of the current instance has an overall positive or negative impact on a sequence of instances total time cost.

5.6.2 SEQUENCE RUNS

The runs show a benefit on both the operational metrics as well the average number of times an order needs to pass through the matching problem for a *shopper* to be allocated to its *assignable*.

The number of times an order is a candidate for allocation is particularly relevant for determining if the order will arrive within the promised delivery window.

Orders arriving late are affected by many factors other than allocations, such as shopper know-how of stores, difficulty of delivery, traffic changes due to external factors, the correct calculation of elapsed times, etc.

Although not explicitly considered when matching for this thesis, as the *pipeline model* does not do so, complying with the delivery windows is seen as the main factor related to customer experience, therefor a very important factor for order dispatch.

Looking at the results for Zone 4 and Zone 3, the actual impact of switching the matching engine varies from zone to zone, from approximately 0.9% worse efficiency to just over 22% less total time cost, with the best performing model of the proposals, which for operational metrics is the *threshold model* on both cases.

Table 11 | Run results total time cost summary performance

Result	Pipeline model (minutes)	Threshold model (minutes)	Difference	Optimal allocation model (minutes)	Difference
Zone 3 – Morning	27.7	24.4	-11.9%	25.1	-9.5%
Zone 3 – Noon	25.3	23.8	-5.9%	24.8	-2.0%
Zone 4 – Morning	30.3	23.4	-22.8%	25.8	-14.9%
Zone 4 – Noon	31.6	28.7	-9.2%	30.3	-4.1%
Zone 1 – Morning	29.7	28.9	-2.7%	30.8	3.7%
Zone 1 – Noon	32.1	32.4	0.9%	32.7	1.9%

For context, if the time savings shown for the best-case scenarios of each zone were directly transferable to operational cost[17], then switching the matching model from the currently used heuristic to the *threshold model* would result in per-order profits improvement ranging from 12% to approximately slightly over 40% depending on the city considered and its characteristics.

It is worth highlighting that the result for Zone 1 on all the mathematical optimization proposals does not show an improvement in operational metrics over the company heuristic, however as

[17] shopper commission plans are distance based, rather than time based which is the optimized metric for all the matching models.

with the characteristics of the single run instances, the scenario for this run is "shopper constraint", which could be hiding future inefficiencies of the company approach during future runs that due to technical limitations of the evaluation implementation can't be properly evaluated on longer time frames.

Amongst the proposed mathematical optimization models, the *threshold model* is the one that achieves the least costly *total time* and has a coverage that although worse than the *optimal allocation model*, is generally better than the *pipeline model*. This emphasizes the previously mentioned conjecture that future projects focusing on the non-allocation decision could yield even better performance.

The main drawback of the proposed models is in the fact that when under *shopper* constrained scenarios, they sacrifice mean allocation costs for greater match coverage with the tested parameters.

The mathematical optimization models achieve greater match coverage, however from the results shown only for "*shopper* abundant" scenarios it outperforms the *pipeline model* regardless of metric.

The higher count of allocation tries for the *pipeline model* is generally considered an undesirable factor, whoever the precise effect of missing an allocation depends on several factors, many of which are out of the scope of matching which makes its impact difficult to quantify for a proper evaluation of the tradeoff with match cost efficiency.

5.6.3 CITY MODEL

As expected, the removal of the subdivision constraint from the city-wide runs, also translates into greater allocation metric gains.

Although whole-city models result in larger instances and therefore longer compute times, the bulk of processing time is spent on the pre-processing phase, which is highly parallelizable.

From a variant of the proposed mathematical optimization models implemented by the company, the solve time spent in solver workloads is a small proportion of total compute time, which with an adequate parallelization implementation should yield solve times within the current 30s benchmark even for considerably larger instance sizes, such as a whole-city instance.

In case the workload exceeds the permitted response times for the shopper matching problem engine, a temporary problem subdivision for load balancing could be added to improve the max solve time, only while the city model instance becomes too large and reverting to the city-wide proposal once the load is stabilized.

The possibility of solving larger instances by removing artificial instance partitions would compound on the benefits of switching to a mathematical optimization approach.

This is just one example of how a mathematical optimization model provides benefits not only AS-IS, but also permits the implementation of techniques that previously were either hard to achieve or simply unfeasible with the *pipeline model*.

6 CONCLUSION

The implementation of a mathematical optimization model for the *shopper* allocation problem in a distributed gig-economy logistics application, shows to generally be more efficient than the current approach used by the company. There are some scenarios where this is not the case, but this can be explained by the technical limitations of the implementation used to produce the results shown in this thesis.

Although promising, the sample of results shown does not allow one to statistically prove the benefits of switching to a variant of the proposed models. A full evaluation of the proposed model would require analysis of statistically significant sample size for both the proposed model and the *pipeline model*.

The conclusions presented in this thesis are there for to be taken as conjecture. However out of the scope of this thesis a production version of a variant of the *optimal allocation model* was tested by the company and although the size of the improvement does not match the one presented in this thesis, both this controlled trial and the implemented model achieve an improvement over the company model.

6.1 FURTHER DISCUSSIONS

6.1.1 NON-ALLOCATION COSTS IMPROVEMENTS

The cost associated with the variable Y_a is a new concept introduced with the proposed formulation. This cost clearly has an impact of the greediness of each instance allocations as shown with the proposed *threshold model*. When the non-allocation cost is set to a smaller value the performance of the model yields more efficient results with less drawbacks compared to the *pipeline model*.

For this thesis the value was set as a fixed cost across instances and a fixed value depending on assignable type (for batch *assignables* a larger value is used). These two simplifications are a reasonable approximation of the behavior of the company heuristic, however further improvements could be achieved by allowing more customization of the associated non-allocation cost.

Heterogeneous non-allocation costs

The company approach includes implicit preferences in the search for optimal *shoppers* by adjusting the order in which *assignable* are evaluated, as is, this “feature” is not incorporated to any of the mathematical optimization proposals, however heterogenous non-allocation costs can help set these priorities, particularly for shopper constraint scenarios.

It can be argued that the sorting mechanisms used in the company approach also play a larger role for shopper constraint scenarios, as shopper abundant ones will have efficiency metrics play a larger role since, they are part of what can be described as the heuristic objective function.

Dynamic non-allocation costs

The *optimal allocation model* for the sequence runs consider static non-allocation costs, where the deferral of a match for a particular assignable is indifferent to the amount of time that assignable has spent trying to find a shopper match.

In an implementation of one of the proposed models, the preference for previously non-allocated *assignables* can be set with a higher non-allocation cost. The preference for previously un-allocated *assignables* should not affect efficiency metrics for instances with a healthy *shopper* to *assignable* ratio, as in these cases its very likely that all *assignables* will end up with an allocated *shopper*.

For *shopper* constrained instances, preferring the allocation of a slightly less efficient match to avoid late order arrivals is a prefer method from an experience quality point of view, if the costs associated with this tradeoff is a business decision, not a technical one.

A more "organic"[18] use of dynamic non-allocation costs would be an explicit cost for on-time orders within the formulation objective function (7). As previously mentioned, the fulfillment of orders within the promised delivery window is currently the responsibility of tools that assume a somewhat deterministic behavior of *shopper* allocation models.

The reliance on idealistic outcomes from the *shopper* allocation algorithms for controlling on-time order arrival leaves a sizable opportunity for improvement of the customer's experience. Adding these controls within the matching tools enables using more information for this purpose which in theory should reduce late or early orders.

The addition of an on-time associated cost may have a large impact on the logistics cost components of the model. The evaluation of adding this feature is both not a part of the current model (there for, out of the scope of this thesis) and heavily depends on the output of internal predictor tools, making the development of this feature dependent on the work of other teams, this lends itself better to a company project than a thesis.

6.1.2 DELAYED ORDER ALLOCATION

From a matching point of view, it is clearly preferable to find an allocation requiring as few tires as possible, since this should allow for less chance of an order arriving late. The impact of the amount of time required to find a match for an order is not straightforward, since as mentioned the costs are not easy to determine due to differences of the perceived value of arriving on-time for each customer.

The operational impact of not managing to allocate an order has both a direct economic impact (shipping costs refunds are issued under some scenarios) and implicit economic costs, as arriving after the promised delivery window degrades the perception of the service delivered by the company, which leads to a higher chance of losing the customer who placed that order.

[18] Organic use: the proposed implementation aligns with the driving factors for on-time orders rather than a proxy

The expected cost for losing a customer due to poor experience with orders can be obtained using the *customer lifetime value* for that given market multiplied by the probability of losing a customer due to a re-scheduled order, however this last component is not easily obtained, since customers have unique needs for a given order.

Missing the order with your child's birthday cake will not have the same impact as a box of chocolates, unless your partner is in an unbelievably bad mood, then the chocolates might be an urgent matter.

The direct cost of missing the promised delivery window is easy to calculate, as refunds are issued when orders arrive late (and only for late orders, early orders do not trigger a refund), however the refund issued is only triggered for orders that arrive after a given threshold that can be adjusted for each market where the order takes place.

Aside from being market dependent, the fixed cost is a binary decision on whether the "permitted lateness" was reached or not and makes no distinction between orders that arrived a minute later that the threshold from those that arrived hours later.

The amount issued in late arrival refunds is equivalent to the cost of a paid delivery on the platform, which from a costs point of view could be considered as the logistics cost of an average order, this cost is primarily dependent on the characteristics of zone where the customer orders most often.

A limitation of using 30 minutes runs for the impact analysis is entire day consequences can't be adequately modeled. The largest penalties for the untimely allocation of a match arise when merchants close the respective *branches*, this is not captured by a run that does not cover ed of day circumstances.

The consequences of having a larger amount of un-allocated *assignables* is hard to fully quantify without simulating complete day runs, which is not shown due to code performance issues.

The impact of the switch to a mathematical optimization model is there for, greater than the result shown in this thesis, although judging from the *assigned allocation tries* results, it is expected for the difference to be in favor of the mathematical optimization proposals versus the *pipeline model*.

Undelivered orders

A particular case of late orders is those that end up being not delivered within the same day that the order was scheduled for. From an operational point of view these are rescheduled for the next available unless the customer requests a cancelation or another delivery time that is not the earliest available.

As mentioned in 6.1 the control for delivering orders within the promised time window is not directly considered within the scope of shopper allocation problem, however it is considered for the creation of instances.

Instances are created with *assignables* that contain orders for which the estimated elapsed time and the current time is within the orders promised time window. The process of controlling for an on-time delivery assumes that the assignable is allocated within a negligible amount of time for the total order cycle time.

From the simulation results shown in this thesis, some *assignables* can take considerably longer to match than the typical order, particularly ones that are harder to match. It can be easily identified that although for an average case the allocation time might be small, under some circumstances this time metric may be considerable for an assignable.

6.1.3 FAIR SHOPPER SELECTION

From the results shown for both single instance and sequence runs, it can be easily identified that the proposed implementation for a shopper fairness criterion has a large trade-off between operational cost (total time cost) and rewarding shopper connected time.

From the results shown, the proposed fairness heuristic perform worse (and considerably so) than the *pipeline model*, which negates any advantage of switching to a mathematical optimization formulation.

As described in 3.1, the implementation of this feature for the *pipeline model* is done only for match candidates that are within a set a threshold of the most efficient match for each assignable. This notion of only considering a neighborhood of efficient match candidates differs from approach shown, of having the *shopper* wait time be an additional cost for the matches of that *shopper*.

A fairness selection heuristic that more closely resembles the neighborhood approach used in the *pipeline model,* can be reasonably achieved with a mathematical model formulation. Most commercial solvers (e.g. CPLEX and Gurobi) allow the use of a solution set for larger problem GAP configurations, where instead of returning a single solution, a set is returned where all solutions within the set are within a problem GAP set by the users.

The “solution set” functionality implemented for these solvers, returns a set of states for decision variables. Every solution within the set of solutions has an objective function cost that is below the optimality gap set for the instance.

A second decision of which solution between the solution set is then defined as the solution that maximizes the *shopper* idle time, there by rewarding shoppers that have been available and idle the longest.

The downside of a GAP approach is that it would require a more robust layer of post-processing on the solution of the first stage, this would introduce additional complexity in that there could be competing interests on the definition of the best solution for the second stage.

Some teams might ask for the solution with the most amount of shopper fairness, while it can also be valid to choose that the maximizes the number of matches or the one that maximizes on-time deliveries.

Another approach for introducing tie breakers amongst *shoppers* would be to use the idleness metrics for a rank cost associated with the feasible matches for that *shopper*. Since candidate matches are pre-calculated a rank could be generated by sorting the feasible matches for a given assignable and then assigning a cost to the rank for each match candidate e.g.: the largest idle time gets rank 1 and no added cost, rank 2 gets a small added cost, rank 3 and so on get progressively larger costs.

This approach introduces both the computational complexity of having to sort a potentially large list and similarly to the *fairness model* results, requires a cost associated with each rank such that the impact to the efficiency metrics is minimized, where there is no guarantee for the results to have a lower penalty for introducing a fairness component as with the approach shown.

A simpler approach would be modifying the approach shown, to remap the *shopper* wait time to an s-curve instead of a linear cost function, which should help to reduce the efficiency impact of unreasonably large handicaps as well as removing some of the impact for *shoppers* with low handicap values.

Using a non-linear cost function for wait times may improve the impact of a handicap for *shoppers* in "the middle of the pack". This approach, as with others previously mentioned, introduces the need for possibly constant tuning of the threshold values for the taper of each side of the remapped s-curve.

The approach could also yield more efficient results and while maintaining its fairness criterion by tunning the portion of the fairness metrics considered for optimization, the extent to which the efficiency vs fairness tradeoff is made is a business decision.

Another interesting proposal that has the benefit or a more intuitive explanation, is to associate a fair selection to an explicit cost associated with a new variable of "no allocation for *shoppers*" which would be analogous to the *non-allocation* variable for *assignables*.

Regardless of which of the discussed avenues for addressing the accumulation of matches in a reduced group of "*super shoppers*" replicates the business requirements more faithfully, the degree to which the sacrifice of operational efficiency is made requires decision making by the operational teams and tunning the impact of the approach to meet the desired level of tradeoff.

6.1.4 CITY MODEL

The results shown by eliminating the geofencing partitioning for cities that are composed of multiple zones, show an improvement on a regular operation instance, however the partition of a larger city also plays a role when the operation experiences issues that are localized to a part of the city.

Operational issue isolation

If there is an operational issue in a portion of a city, such as a road being blocked, it's preferable to isolate the affected area allowing the remainder of the operation to continue to provide matches if available.

A solution for issue isolation could be a manual approach for temporarily generating geo-fenced excluded sections of the city, however this requires a monitoring team and the team following established procedures, ideally avoiding the unintended use of such functionality.

The approach of requiring team input for problem identification is not scalable for many operating cities, however due to the very nature of these situations being "freak events" it can be safely assumed that the workload of identifying these events and taking action should not present a major problem for the operation, however days with stressed operation could present a problem.

Shopper imbalance

Another potential issue introduced with eliminating the currently used approach of subdividing cities lies in the potential drift of *shoppers* across what are currently treated as different subdivisions.

By removing a way of avoiding *shopper* drift, the potential for imbalance between *shopper* and demand locations is greater than with the current approach. For cases when there is greater demand imbalance the cost of matching these scenarios could negate the benefits of solving the problem without the subdivisions.

How *shopper* behavior may change with the removal of city partitions for matching is hard to estimate from an offline approach and should be tested by the company. *Shoppers* may react and coordinate (app-based contractors commonly communicate through messaging platforms and social networks such as Facebook and WhatsApp groups) to what they perceive as a change in demand which might not match the operation.

Shopper preferences are also not considered for this thesis, which could also lead to unexpected results when implemented. *Shoppers* might prefer working in a subset of neighborhoods due to traffic, proximity to their household, safety, etc. How these preferences may impact the mentioned drift is not within the scope of this thesis but should be considered for the implementation of solving whole city models.

Given the contractor nature of shoppers, there are little legal avenues for addressing this issue, and although it is also a potential problem with the current geofenced cities it is exacerbated by eliminating these constraints.

Amongst the feasible tools that could address this issue are pricing solutions for *shopper* re-distribution as well as naturally having demand that requires a more even distribution of the locations for shoppers.

Pricing and information distribution tools, such as a demand forecast, could by providing value for the *shopper* that does not drift to another neighborhood, reduce the benefit that each individual *shopper* has for drift and the large-scale consequences of groups of *shoppers* behaving in this way.

Regardless of the solution and its impact, the redistribution of shoppers with this proposal should be monitored and addressed if it becomes a problem for the operation.

6.1.5 RUNTIME AND ENGINEERING ASPECTS

Although not discussed in depth, the runtimes for a mathematical optimization implementation for the *shopper* allocation problem must run within the same or better max runtimes as the current the heuristic used by the company.

For practical purposes the workloads can be separated into:

- pre-processing
- shopper selecting

Pre-processing

The pre-processing component is done in python 3.8 for this thesis, which is also the programing language used by the company. The main difference between the heuristic and all proposed models is that the mathematical optimization approach does not require the performance improvement steps used in the *pipeline model* as for the proposed models.

Artificially reducing the match candidate count could result in resource competition, which is an unnecessary problem if code performance is within the required running time benchmark (some steps of the company heuristic are only aimed at improving runtime performance).

The compute times for the preprocessing workload should be very similar between the heuristic and the proposed models, with the mathematical optimization models having slightly longer runtimes for large instances, since more match candidates are evaluated than for the *pipeline model*.

An important benefit for the mathematical optimization approach is that it eliminates the dependencies across business checks for different *assignables*, since the allocation is performed in bulk with the formulation presented in chapter 4.2

By having the process of applying business filters independently amongst *assignables*, this workload can be scheduled in parallel for an instance, instead of the current requirement to process these filters in series.

With the use of large multicore compute nodes that are readily available on most cloud providers, parallelizing the pre-processing workload should allow for solving much larger problems that the current approach is able to do within the same benchmark times set for this workload.

Shopper selection

The shopper selection process requires more steps for any of the mathematical optimization proposals, as a model need to be built, compiled, solved, and read, whilst the company heuristic uses python built in loops and object to select the best shopper.

This means that for smaller instances, the mathematical optimization proposals will take longer than the company approach. For large instances the solve times should be shorter than the heuristic. From the runs of this thesis the formulation was solved in less than the benchmark time for an instance that is roughly 3 times larger than the peak instance size of the regular operation.[19]

It is worth noting that a modified version of one of the proposed formulations was tested within the operation using open-source solvers and python compilers. For this production version of the mathematical optimization models both the runtime and memory use were well within the acceptable benchmark times for the same instances using the company heuristic.

In summary the deployment aspects for the proposed models should be slightly worse than the current model if no parallelization is used, but still be within the benchmark runtimes for this application. For larger instances a parallelized version of the pre-processing workload can be implemented which would allow for solving much larger problems than what is currently possible in the same timeframe.

[19] The model was solved using a 4.4Ghz, 16 core AMD processor with 64 GBs of ram, the entirety of the shopper selection workload is single core while the pre-processing used a basic parallel implementation.

6.2 PROPOSALS FOR FUTURE WORK

The following proposals are out of scope of this thesis; however, they are enabled by the use of a mathematical optimization approach for the *shopper* allocation problem or become easier to implement than with the current company heuristic.

6.2.1 EXPLICIT ON-TIME ARRIVAL COSTS

The most self-explanatory of the proposals, the importance of fulfilling the promised delivery window is mentioned several times in this thesis, however as previously mentioned it is not currently a part of the objective function of the company heuristic.

A benefit of the proposed formulation for the mathematical optimization models is in the explicit decision for not allocating an order. As mentioned in the discussion of *Dynamic non allocation costs*, the preference for deferring an allocation when it would arrive early and likewise the prioritization when it will arrive late can be made explicitly with the cost for non-allocation in (7).

The introduction of on-time related costs and non-allocation tapering should result in an improvement on on-time arrival. The impact on efficiency metrics, as with the fair shopper selection heuristic, should be tunned according to the business decision to how much of the efficiency aspect can be traded in for a customer experience aspect.

6.2.2 STOCHASTIC OPTIMIZATION FOR FUTURE SHOPPER AVAILABILITY

Although not explicitly discussed, the *threshold model* proposal tries to avoid allocating edge cases for an instance, in that matches that are above a the 90th percentile of the historic matching cost.

The notion behind avoiding these extreme value matches implicitly contains the expectation that a lower cost allocation would be available in a future instance. With the approach used there are no order or city characteristics considered to support this assumption.

The drawback of using a "blind" heuristic for skewing allocations to a lower mean match cost is that orders that are inherently expensive, would have a much worse experience using the service. For these large orders, the chance of being qualified over the threshold is higher, which for some markets would mean artificially deferring the allocation of these orders unnecessarily.

Another issue for using this approach is that smaller cost matches are also not considered as inefficient even though they could be allocated to an expensive match for the characteristics of that order and scenario.

The contribution of adding a stochastic formulation to account for the availability of future shoppers would allow the deferral of a match under the uncertainty of future shopper availability. By adding a non-deterministic approach, it would allow the current instance to account for the chance of not finding a better match during subsequent runs of the matching formulation.

6.2.3 REDUCED RUN FREQUENCY

Although not directly attributed to the results shown, the work realized for obtaining the results shown in this thesis uncovered a greater proportion of instances with either trivial or no feasible combinations.

The reduction of the frequency with which the problem is run would allow for larger instances in each run. Like the proposal behind the city-wide model, this should result in either an equal or better solution to the one currently obtained with little to no impact on the shopper experience or the customer experience.

The downside of these larger instances is in longer running times, although as mentioned in the *runtime and engineering aspects* (6.1.5) of the discussions segment. The mathematical optimization solutions have enough remaining runtime slack that an increase that is within doble digit percentage points should not be an issue, also less frequent runs could also be interpreted as doubling the performance benchmark threshold.

6.2.4 ONLINE OPTIMIZATION HEURISTICS

The re-allocation of previously matched *shoppers* could allow the operation to re-assign a *shopper* when past allocations were comparatively inefficient to that which is possible with the arrival of new information (connected *shoppers,* new *assignables*).

For a *shopper* that is very experienced and a fast picker, may have been assigned a small order because no other feasible options were available, in this same scenario it would be preferable to both the operation and the previously assigned *shopper* for a larger more complex order to be re-assigned to the experiences shopper and its current order re-assigned to another less experience *shopper* if the option were available.

There are both legal and *shopper* experience limitations as to what scenarios are feasible for assignable re-allocation, however the opportunity by definition, is one where there is no cost associated with the re-allocation (unless explicitly implemented to reward a given shopper).

In most other applications of distributed logistics, the opportunity for these re-allocations occurs sparsely due 0 cost options for contractors being less likely. For a ride hailing application riders would have to request the service within close proximity and within a short timeframe for there to be enough cases where applying this proposal would be worth the implementation effort.

The situation is slightly better for meal delivery services, as the demand is concentrated around busy restaurants, however for the groceries and retail delivery service offered by the company larger stores accumulate a large proportion of orders. Given the size of a location of merchants, a small set of locations satisfy a larger proportion of the demand, there for the chance of these win-win scenarios are more common.

The legal and technical limits could still detract from the appeal of implementing a simple notion of online matching, though the little to no operational cost of the proposal should make it worth considering.

6.2.5 FUTURE SCENARIO SETUP COSTS ON CURRENT MATCHES COST

Finally, a proposal that could be considered, is assessing how the current instance's matches would set the availability and position of *shoppers* for instances later during the day.

A limitation of a dynamic programing approach is that the number of combinations that need to be evaluated is drastically larger than the current size of problems solved in company instances, particularly for the matches at the beginning of the planification.

Instances that are orders of magnitude larger than the current problem size could not only affect the solve times but also introduce hardware instance expenses and even limitations on extreme cases.

A potential solution for the problem size issues would be to condense the effect on future instances into a cost within the skew costs of the objective function, in practice this would look like some sort of correlation between shopper metrics and the characteristics of planed orders close by the final destination of the current assignable.

Another possible drawback of this approach is that it does not consider idle shopper behavior. If shoppers perceive that they will have higher earnings by being closer to the origin of the current trip they might disregard any relocation cost and relocate closer to the original destination before another order is accepted.

For shopper behavior considerations a possible competing solution that is delegates the shopper location responsibility to other services within the organization is the use of pricing mechanisms for shopper location balancing during the day.

7 BIBLIOGRAPHY

Abdulkadiroglu, A., Pathak, P., Roth, A. E., & Sonmez, T. (2006, May). Changing the Boston School Choice Mechanism: Strategy-proofness as Equal Access. *Mimeographed. Harvard University*.

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2), 295-303.

Arslan, A. M., Agatz, N., Kroon, L., & Zuidwijk, R. (2019). Crowdsourced delivery—a dynamic pickup and delivery problem with ad hoc drivers. *Transportation Science*, 53(1), 222-235.

Arslan, A., Agatz, N., & Klapp, M. (2019). Splitting shopping and delivery tasks in an on-demand personal shopper service. *Erasmus Research Institute of Management (ERIM)*.

Baker, B. M., & Sheasby, J. (1999). Extensions to the generalised assignment heuristic for vehicle routing. *European Journal of Operational Research,*, 119(1), 147-157.

Bressoud, T. C., Rastogi, R., & Smith, M. (2003). Optimal configuration for BGP route selection. *INFOCOM* (pp. Vol. 2, pp. 916-926). IEEE.

Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science*, 37(12), 1590-1602.

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M.-E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57, pages: 28-46.

Gale, D., & Shapley, L. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly,*, 69(1), 9-15.

Gurobi Optimization, L. (2021). *Gurobi Optimizer Reference Manual*. Retrieved from https://www.gurobi.com.

Higgins, A. (1999). Optimizing cane supply decisions within a sugar mill region. *Journal of Scheduling*, 2(5), 229-244.

LeBlanc, L. J., Shtub, A., & Anandalingam, G. (1999). Formulating and solving production planning problem. *European Journal of Operational Research*, 112(1), 54-80.

Öncan, T. (2007, Agust). A Survey of the Generalized Assignment Problem and Its Application. *INFOR, Vol. 45, No. 3*, 123-141.

Parragh, S. N., Doerner, K. F., & Hartl, R. H. (2007). A survey on pickup and delivery problems. Part II: Transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*.

Roth, A. E. (2008). Deferred acceptance algorithms: history, theory, practice, and open questions. *international Journal of game Theory, 36(3)*, 537-569.

Sahni, S., & Gonzalez, T. (1976). P-Complete Approximation Problems. *Journal of the ACM, 23(3)*, 555-565.

Savelsbergh, M., & Sol, M. (1995). The general pickup and delivery problem. *Transportation science*, 29(1), 17-29.

Srinivasan, V., & Thompson, G. (1973). An algorithm for assigning uses to sources in a special class of transportation problems. *Operations Research*, 21(1), 284-295.

Woodard, D., & Microsoft Research. (2018). Matching and Dynamic Pricing in Ride-Hailing Platforms [video]. *https://youtu.be/cddFAgRyxQo*. Youtube.

Wygonik, E., & Goodchild, A. (2012, September). Evaluating the efficacy of shared-use vehicles for reducing greenhouse gas emissions: a US case study of grocery delivery. *n Journal of the Transportation Research Forum*, 111-126 (Vol. 51, No. 2).

Yu, Y., & Prasanna, V. K. (2003). Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization. *Journal of Information Science and Engineering*, 19(3), 433-449.

Yu, Y., Tang, J., Li, J., Sun, W., & Wang, J. (2016). Reducing carbon emission of pickup and delivery using integrated scheduling. *Transportation Research Part D*, Transport and Environment, 47, 237-250.

ANNEX

ANNEX A - PROPOSED OPTIMAL ALLOCATION MODEL 1

ANNEX A.1 - SETS

Shoppers

$$s \in S$$

Assignables

$$\boldsymbol{a} \in \boldsymbol{A}$$

Stores

$$\boldsymbol{b} \in \boldsymbol{B}$$

ANNEX A.2 - DECISION VARIABLES

Matched found

$$x_{s,a,b} = \begin{cases} 1 \ \textit{if shopper s is allocated to assignable} \text{ a at store b} \\ 0 \sim \end{cases}$$

Non allocation

$$\boldsymbol{y_a} = \begin{cases} \boldsymbol{1 \ if \ assignable \ a \ is \ not \ matched} \\ \boldsymbol{0} \sim \end{cases}$$

ANNEX A.3 - OBJECTIVE FUNCTION

$$\min_{s,a,b} \sum_{s \in S, a \in A, b \in B} x_{s,a,b} \times \left(shopper\_to\_branch\_time_{s,b} + handicaps_{s,a,b} + branch\_to\_custome\_time_{s,a,b}\right) + \sum_{a \in A} y_a \times non\_allocation\_cost_a$$

ANNEX A.4 - CONSTRAINTS

Non allocation activation

$$\boldsymbol{1 - \sum_{s \in s, b \in B} x_{s,a,b} \leq y_a \quad \forall a \in A}$$

Assign at most one assignable per shopper

$$\boldsymbol{\sum_{a \in A, b \in B} x_{s,a,b} \leq 1 \quad \forall s \in S}$$

Store capacity constraint

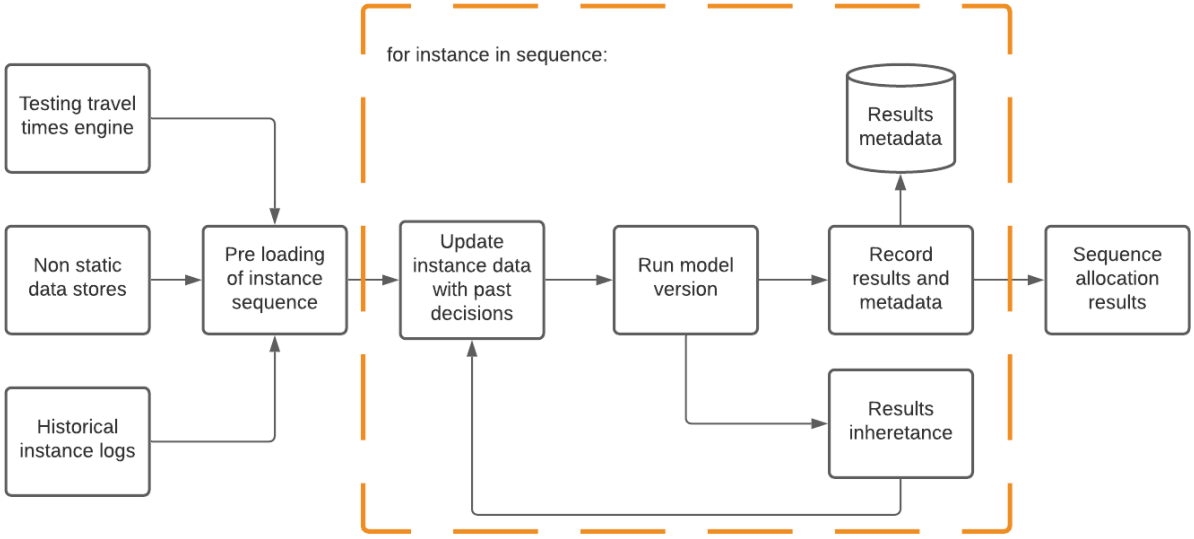
$$\sum_{s \in S, a \in A} x_{s,a,b} + \text{current\_capacity}_b \leq store\_capacity_b \quad \forall b \in B$$

Assign at most one shopper per assignable

$$\sum_{s \in S, b \in B} x_{s,a,b} \leq 1 \quad \forall a \in A$$

ANNEX B - EVALUATION RUNS SETUP

Figure 13 | Run setup depiction



ANNEX C - RESULTS

ANNEX C.1 - SINGLE RUNS

Zone 1

Figure 14 | Zone 1 - Shopper to store travel time performance curve

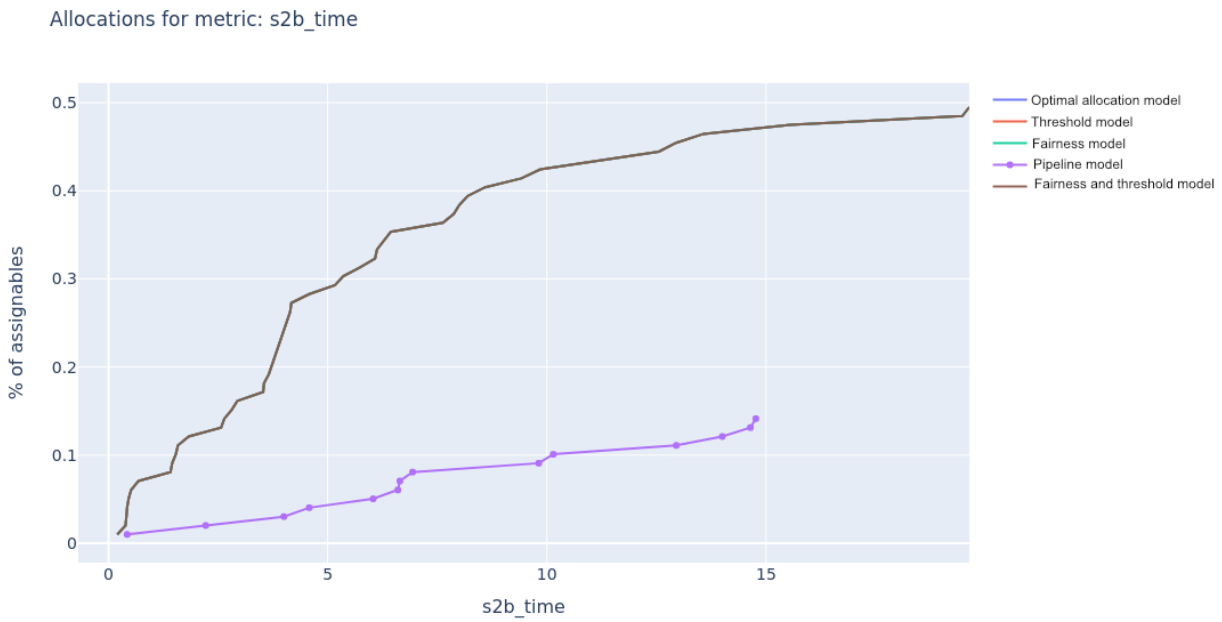


Figure 15 | Zone 1 – Store to last delivery location travel time

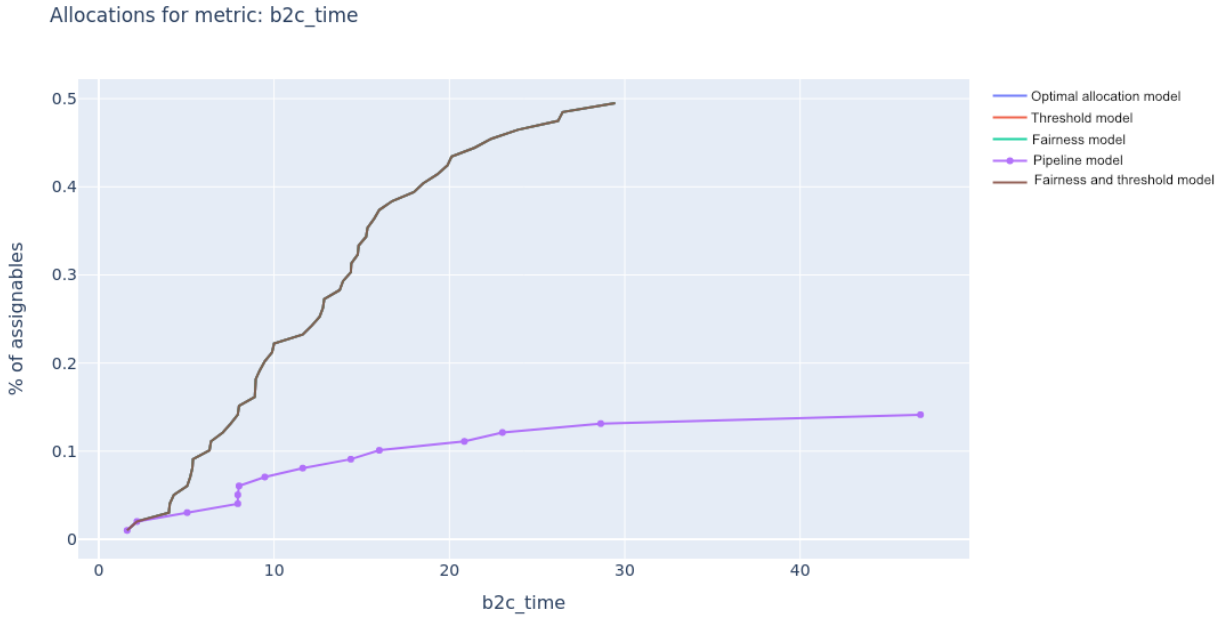
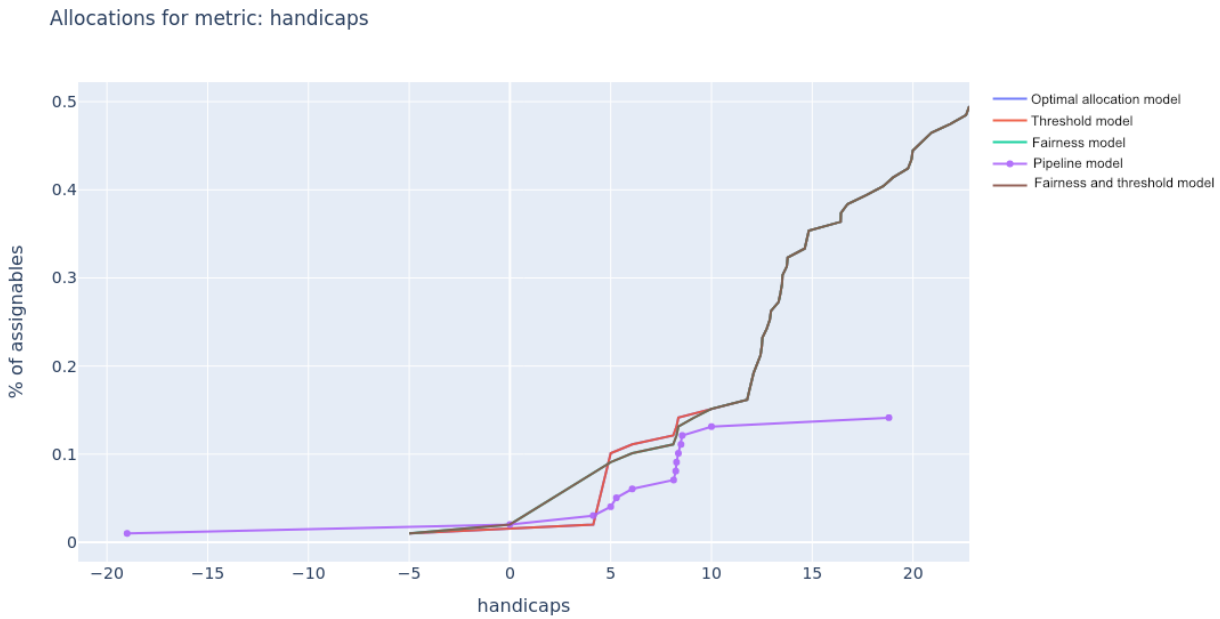


Figure 16 | Zone 1 – business preference skew performance curve



Zone 2

Figure 17 | Zone 2 - Shopper to store travel time performance curve

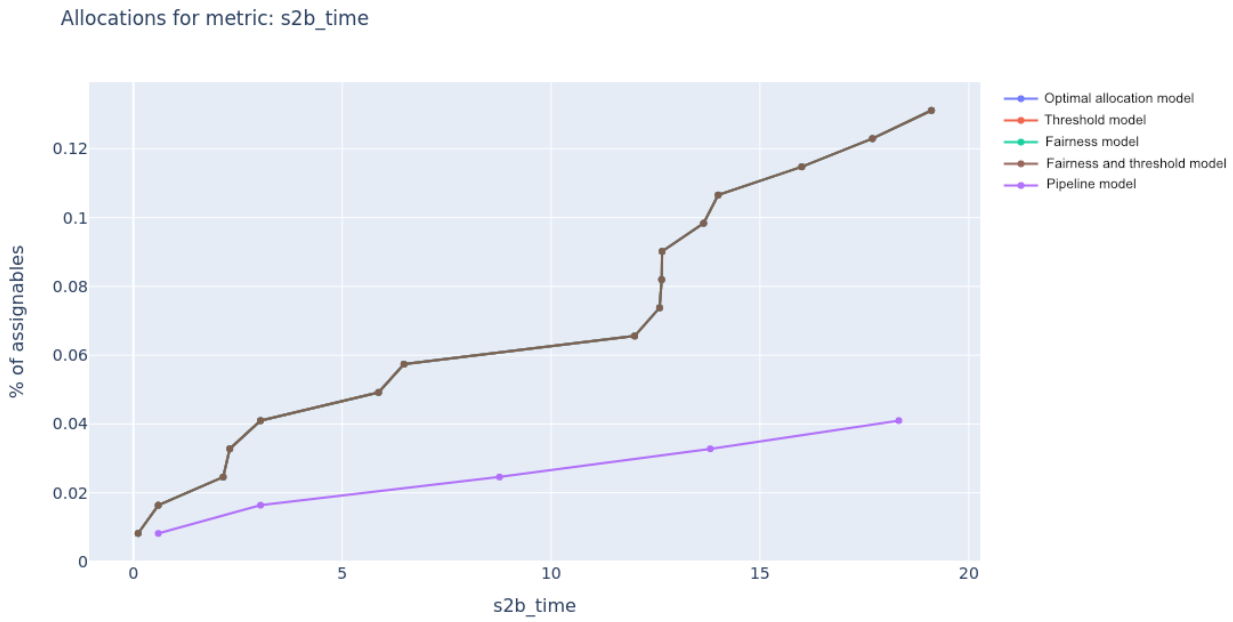


Figure 18 | Zone 2 – Store to last delivery location travel time

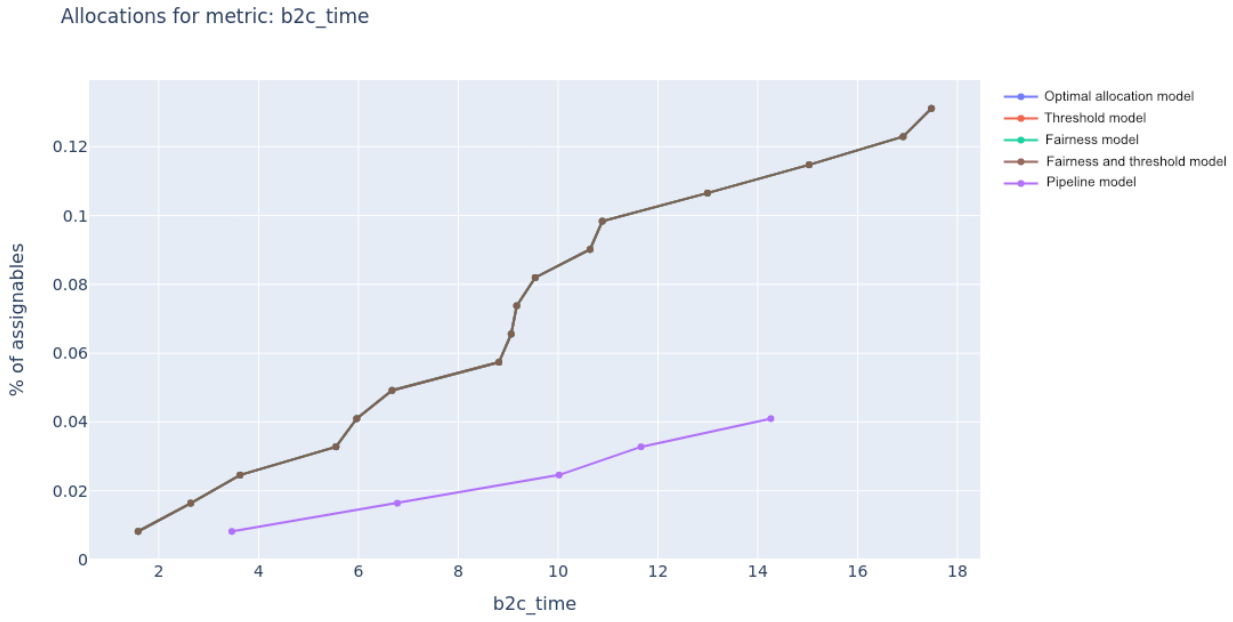
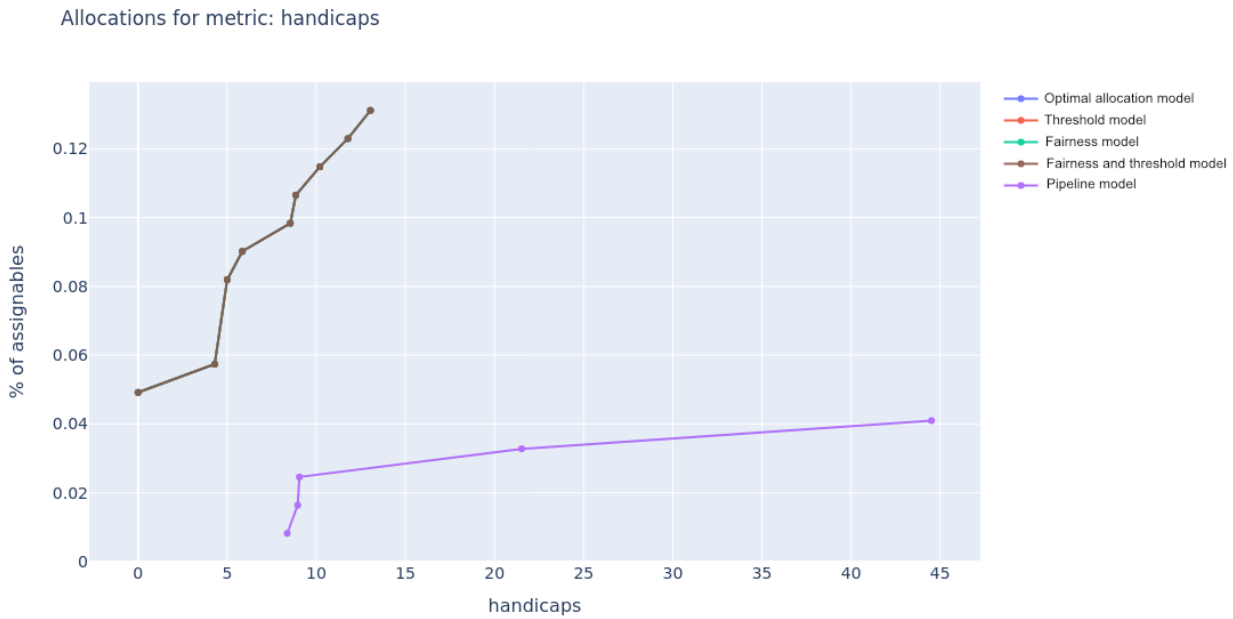


Figure 19 | Zone 2 – business preference skew performance curve



Zone 3

Figure 20 | Zone 3 - Shopper to store travel time performance curve

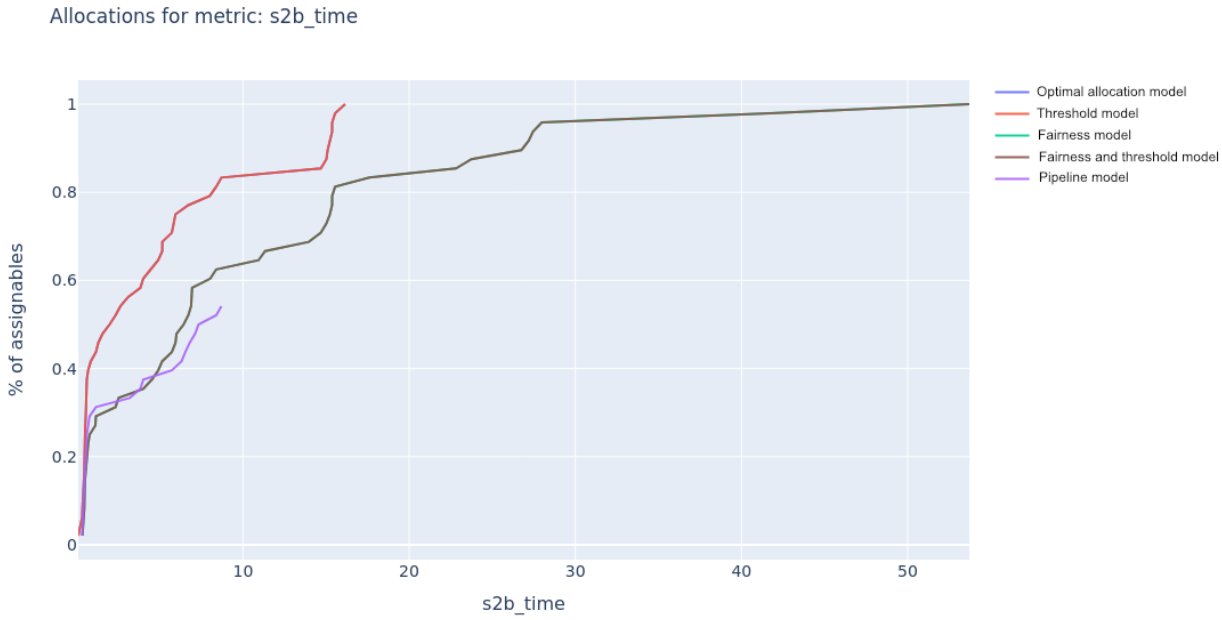


Figure 21 | Zone 3 – Store to last delivery location travel time

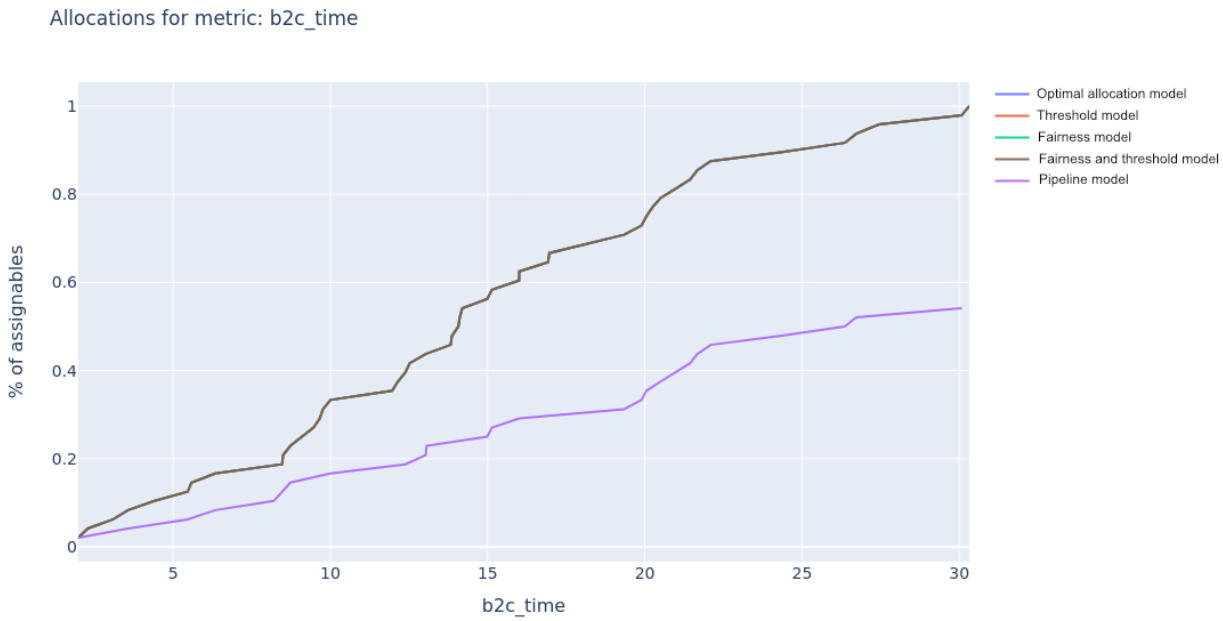
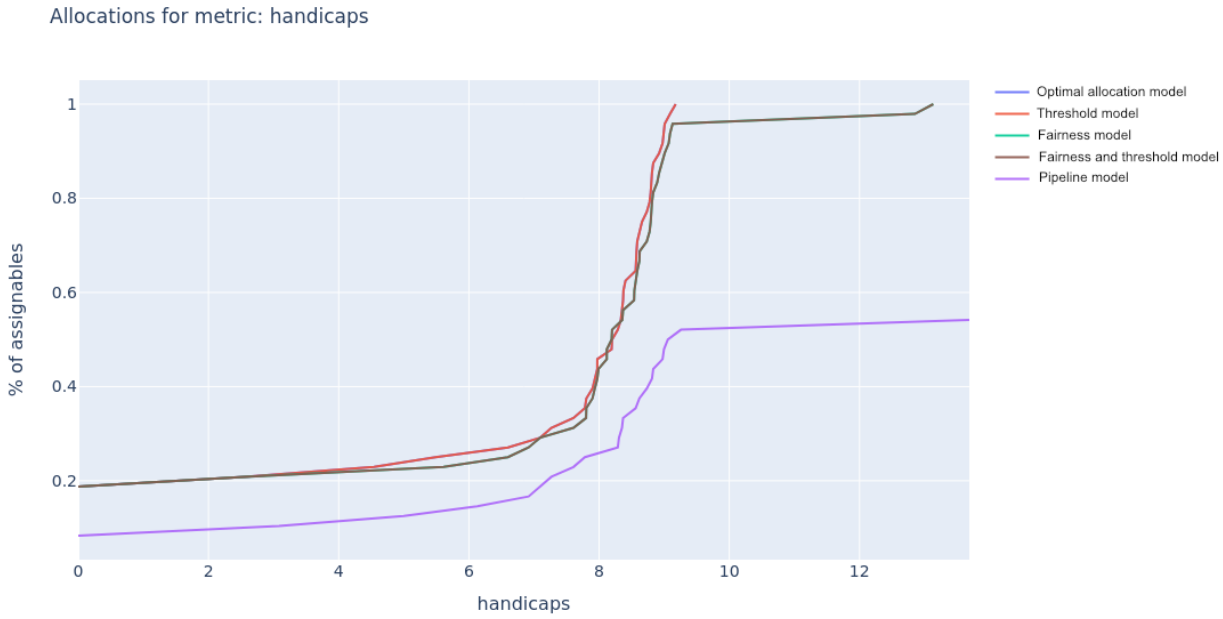


Figure 22 | Zone 3 – business preference skew performance curve



ANNEX C.2 -SEQUENCE RUNS

Zone 3 – Morning run

Figure 23 | Zone 3 - Morning - Shopper to store travel time performance curve

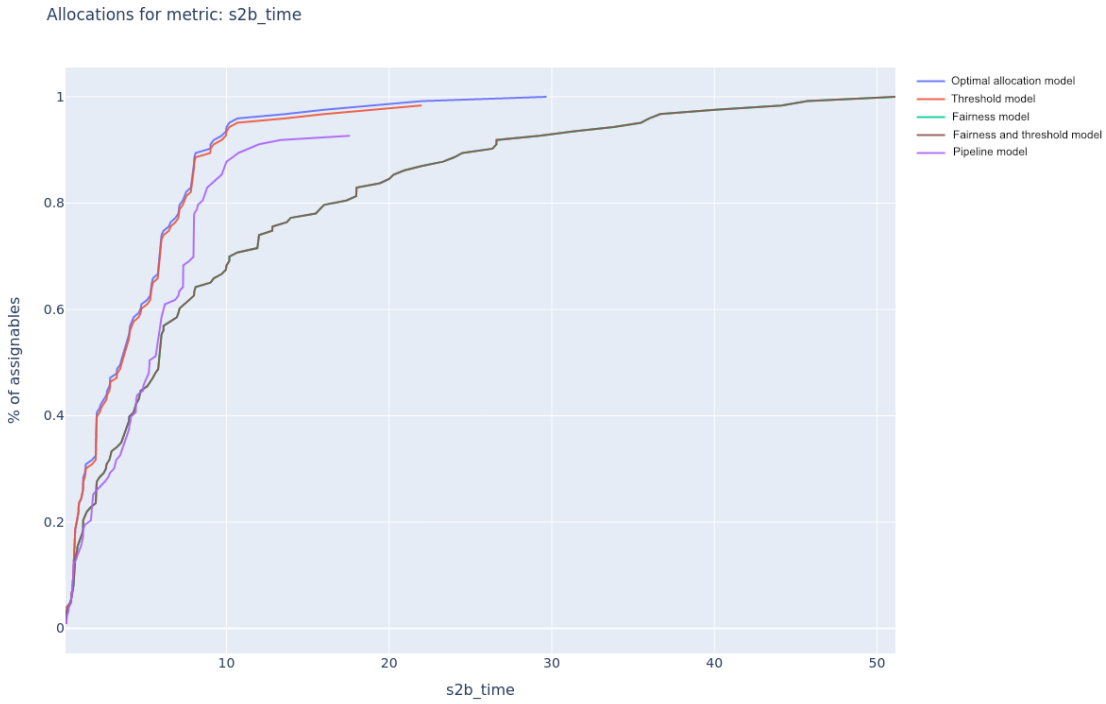


Figure 24 | Zone 3 - Morning – Store to last delivery location travel time

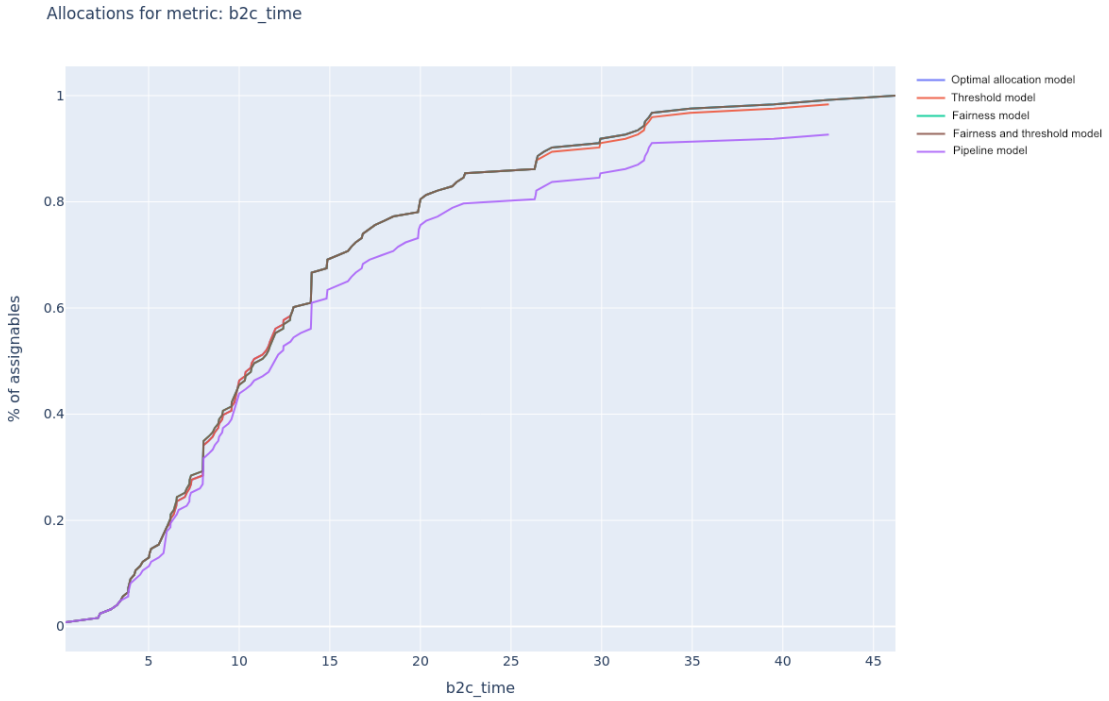


Figure 25 | Zone 3 - Morning – business preference skew performance curve

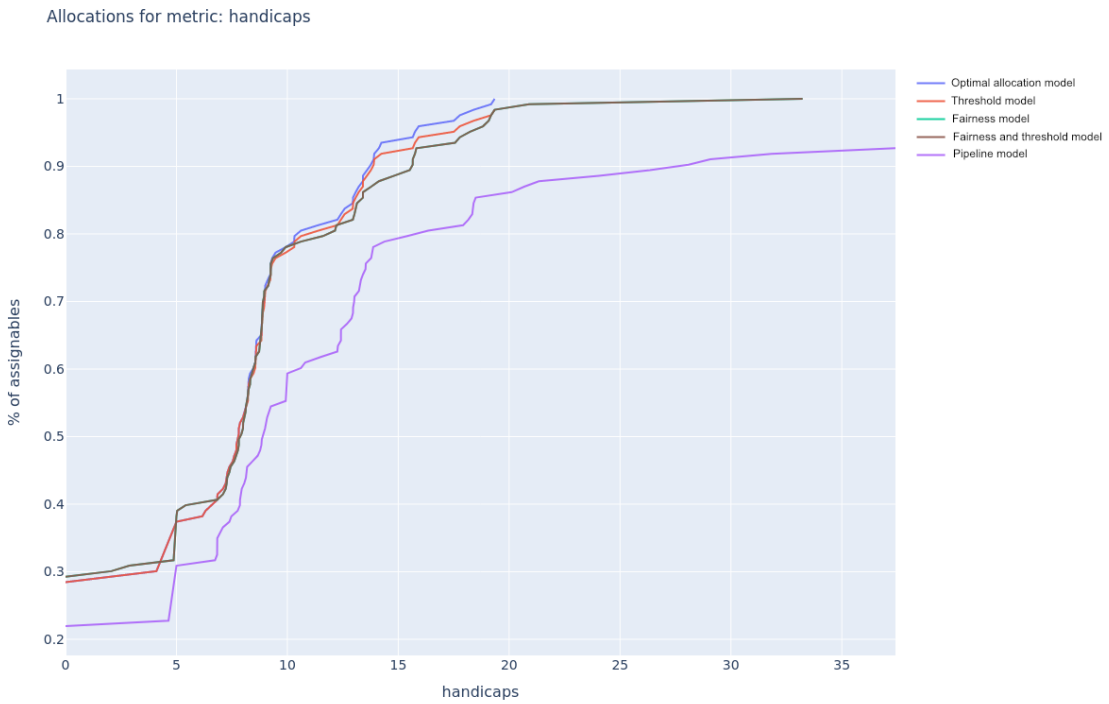
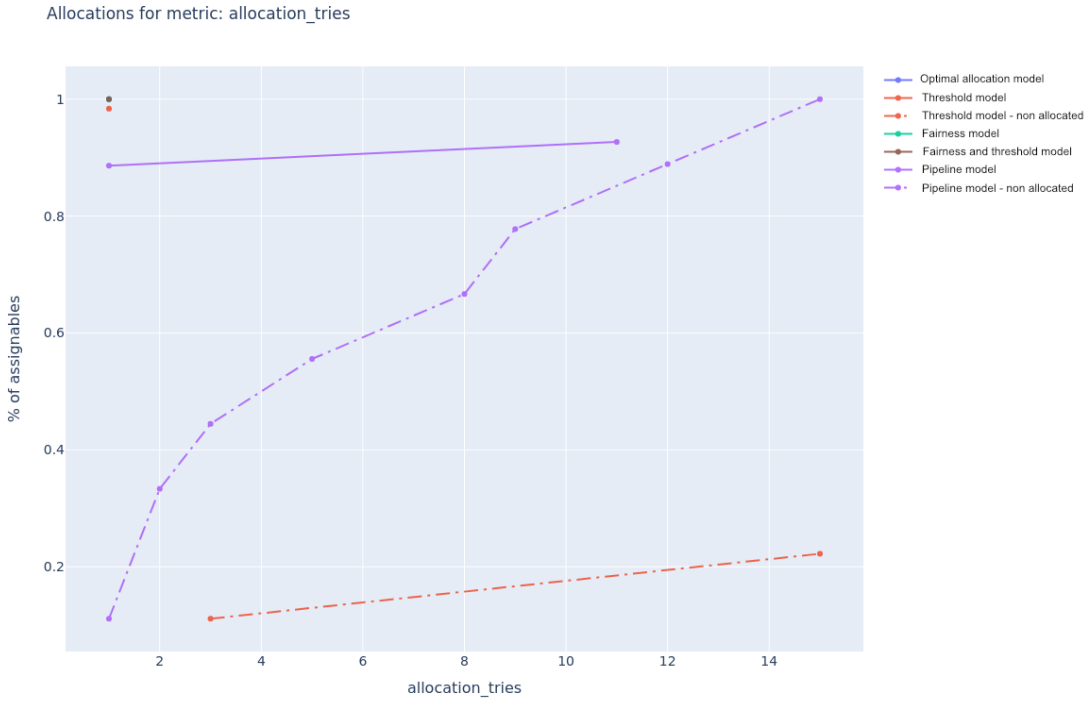


Figure 26 | Zone 3 - Morning – Attempts to find a match tries performance curve



Zone 3 – Noon run

Figure 27 | Zone 3 - Noon- Shopper to store travel time performance curve

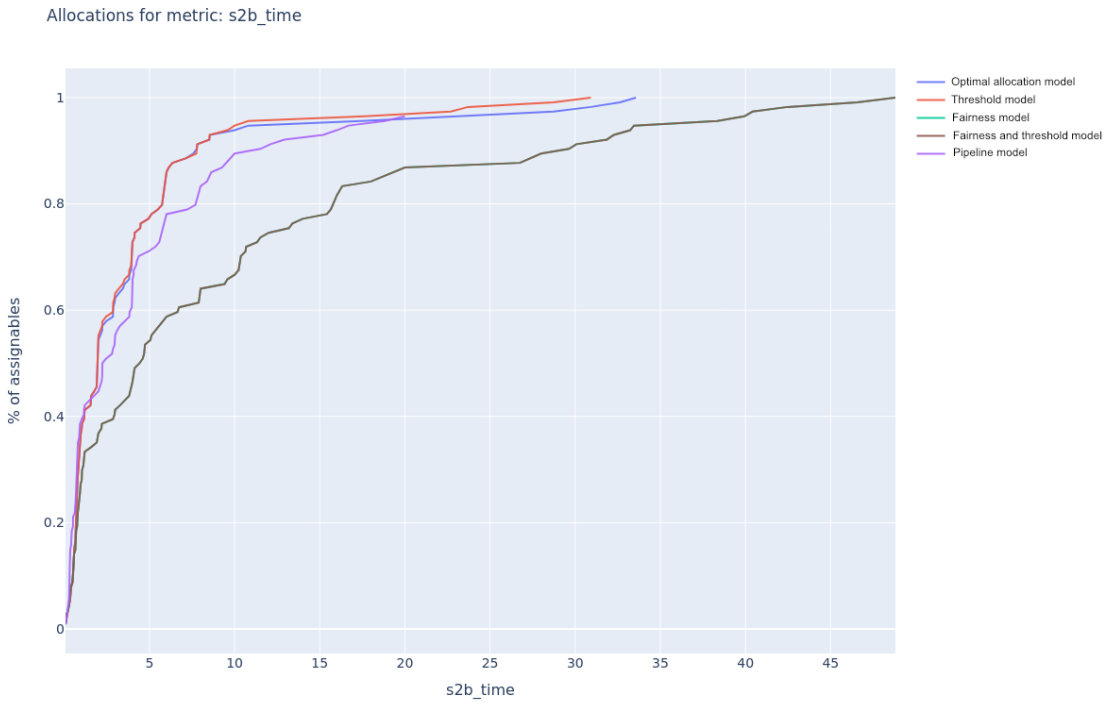


Figure 28 | Zone 3 - Noon – Store to last delivery location travel time

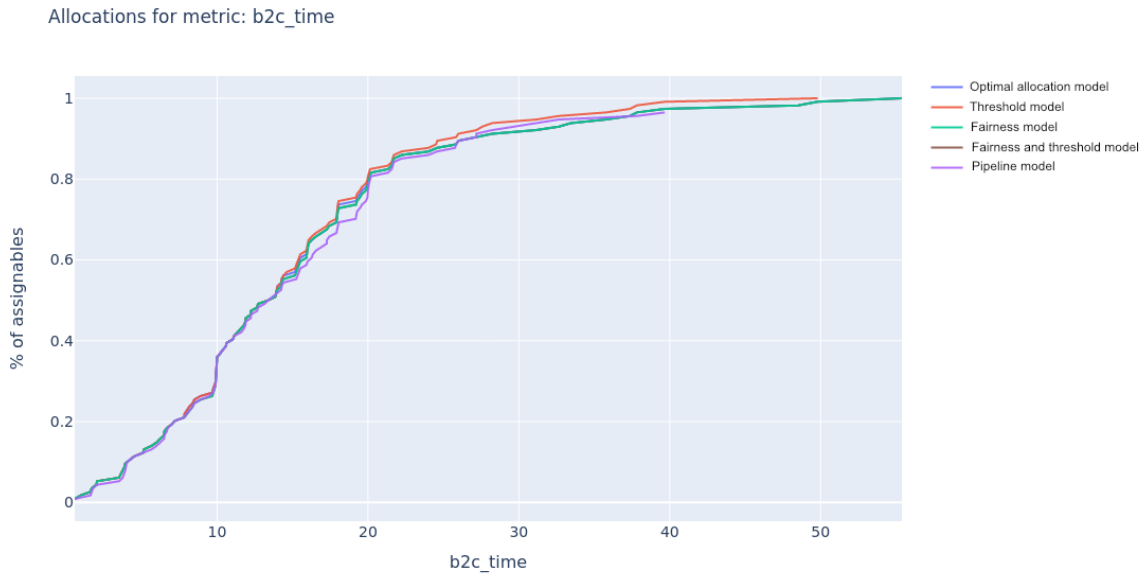


Figure 29 | Zone 3 - Noon – business preference skew performance curve

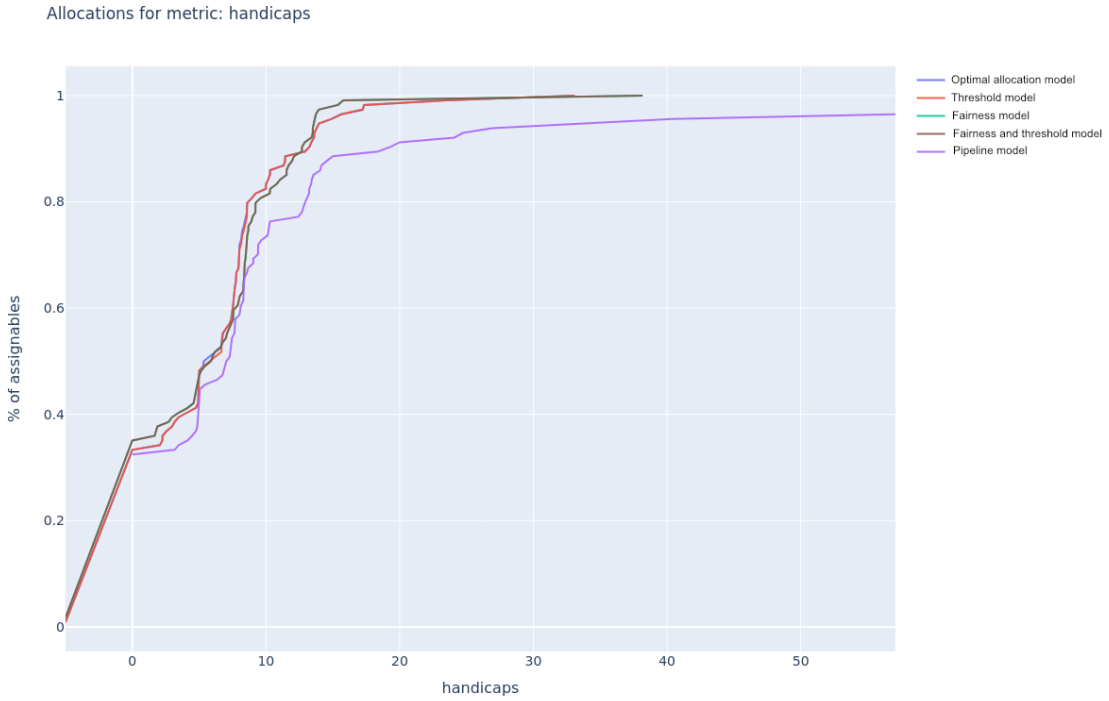
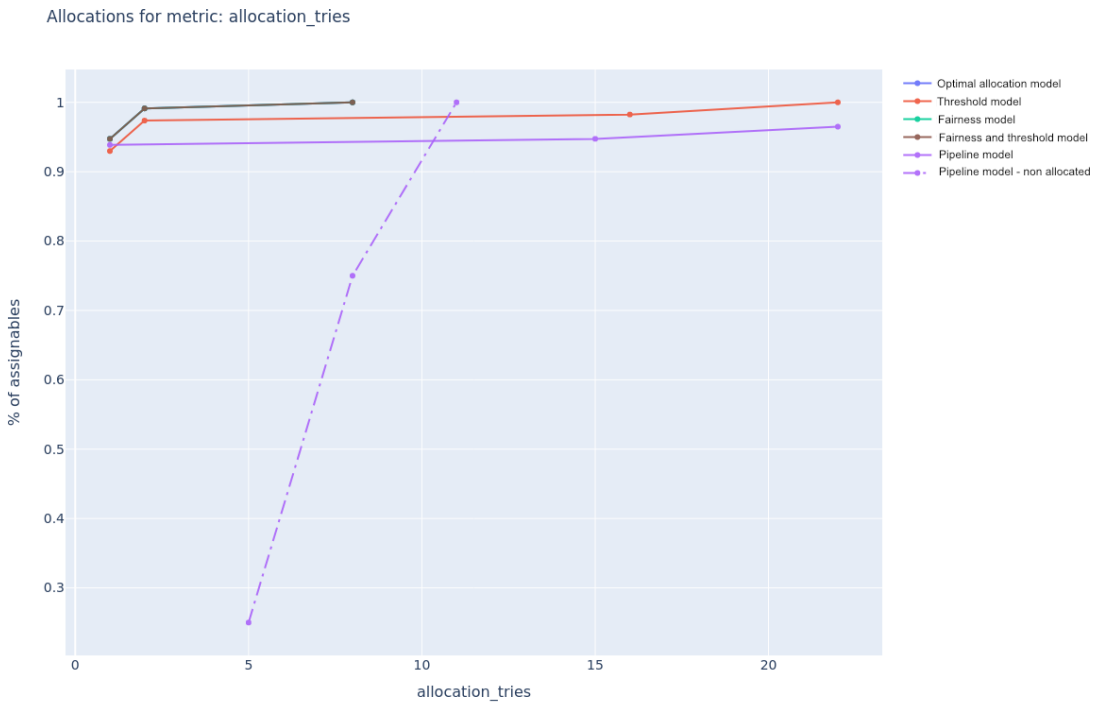


Figure 30 | Zone 3 - Noon – Attempts to find a match tries performance curve



Zone 4 – Morning run

Figure 31 | Zone 4 - Morning - Shopper to store travel time performance curve

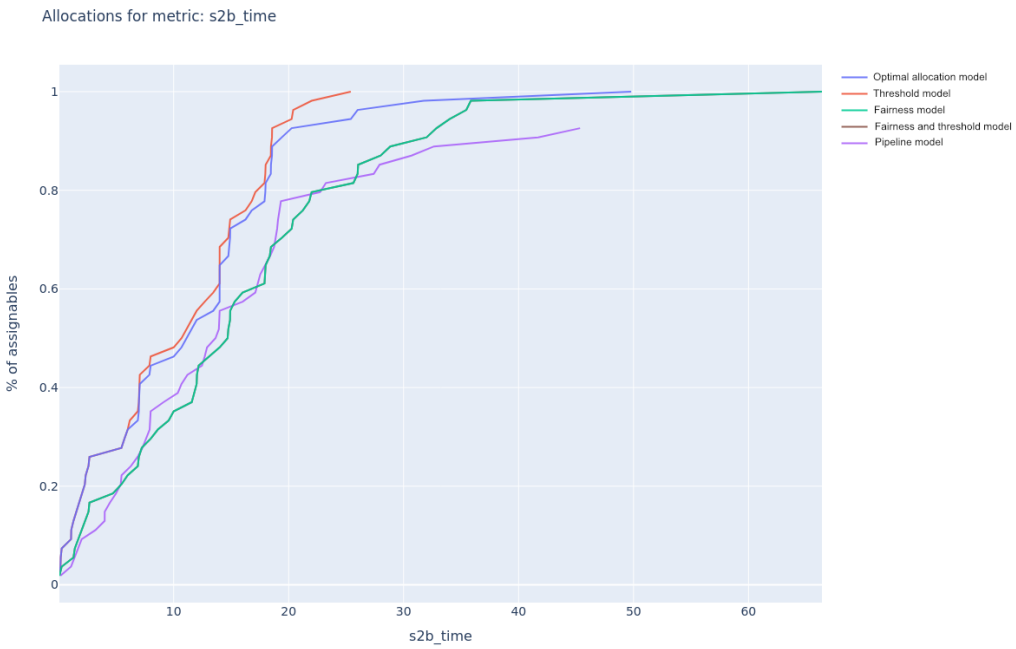


Figure 32 | Zone 4 - Morning – Store to last delivery location travel time

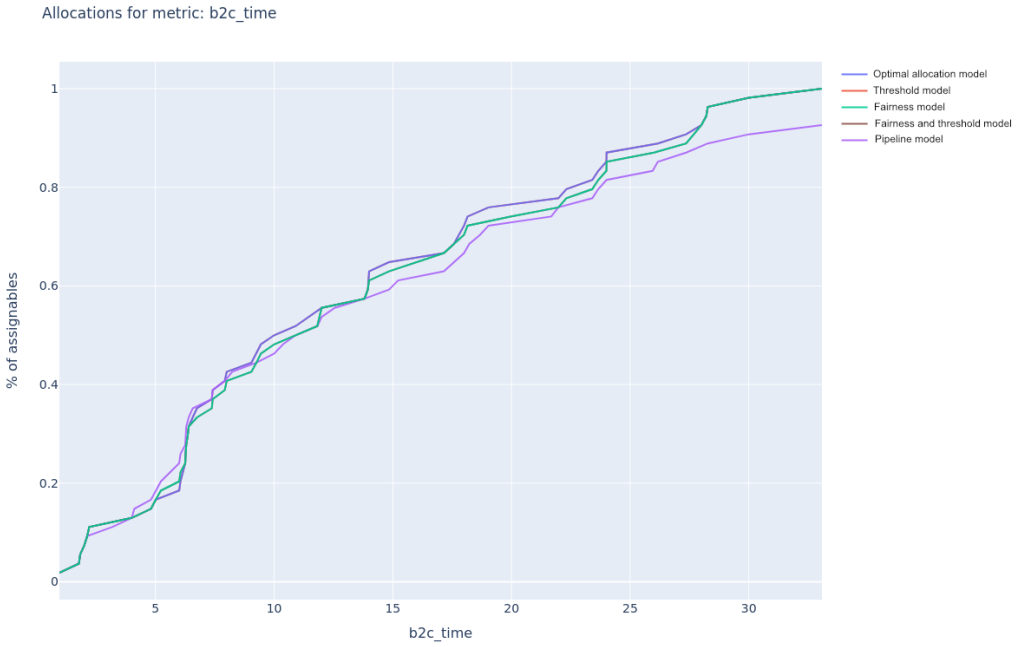


Figure 33 | Zone 4 - Morning – business preference skew performance curve

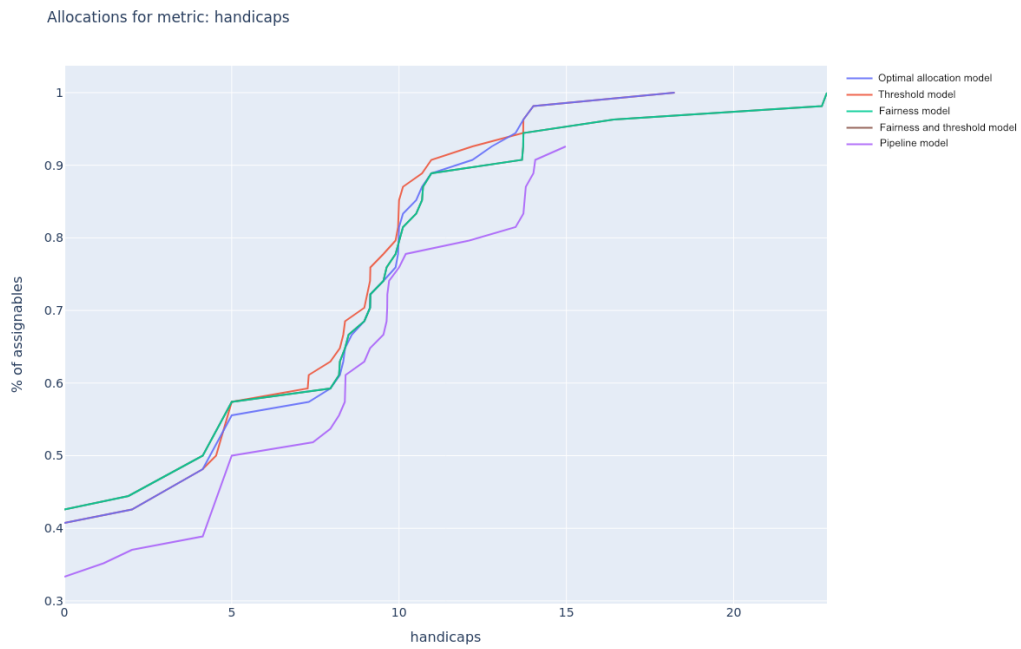
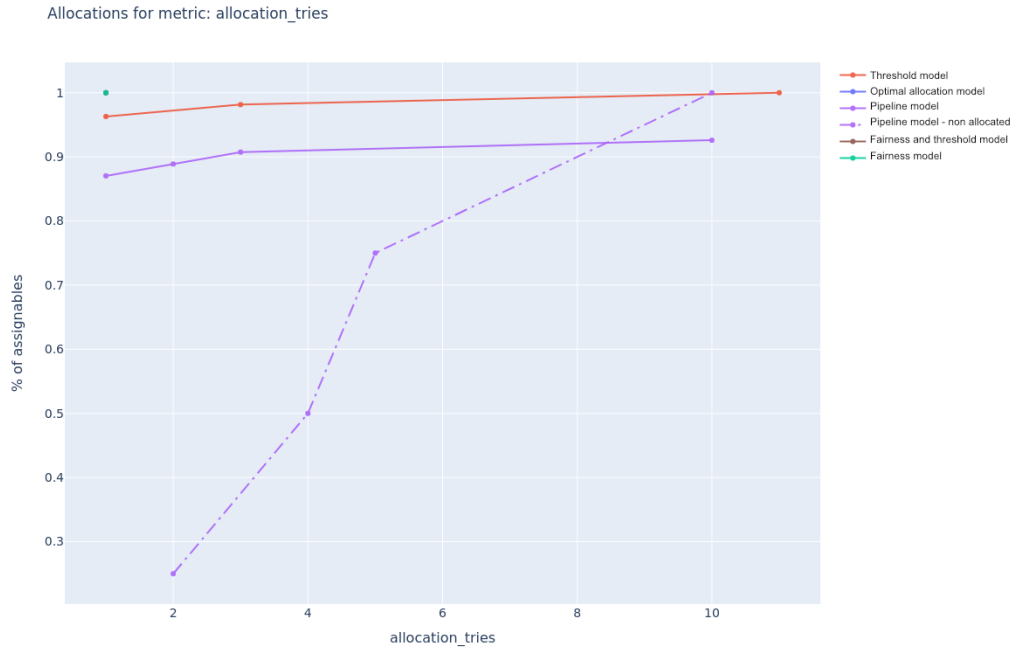


Figure 34 | Zone 4 - Morning – Attempts to find a match tries performance curve



Zone 4 – Noon run

Figure 35 | Zone 4 - Noon - Shopper to store travel time performance curve

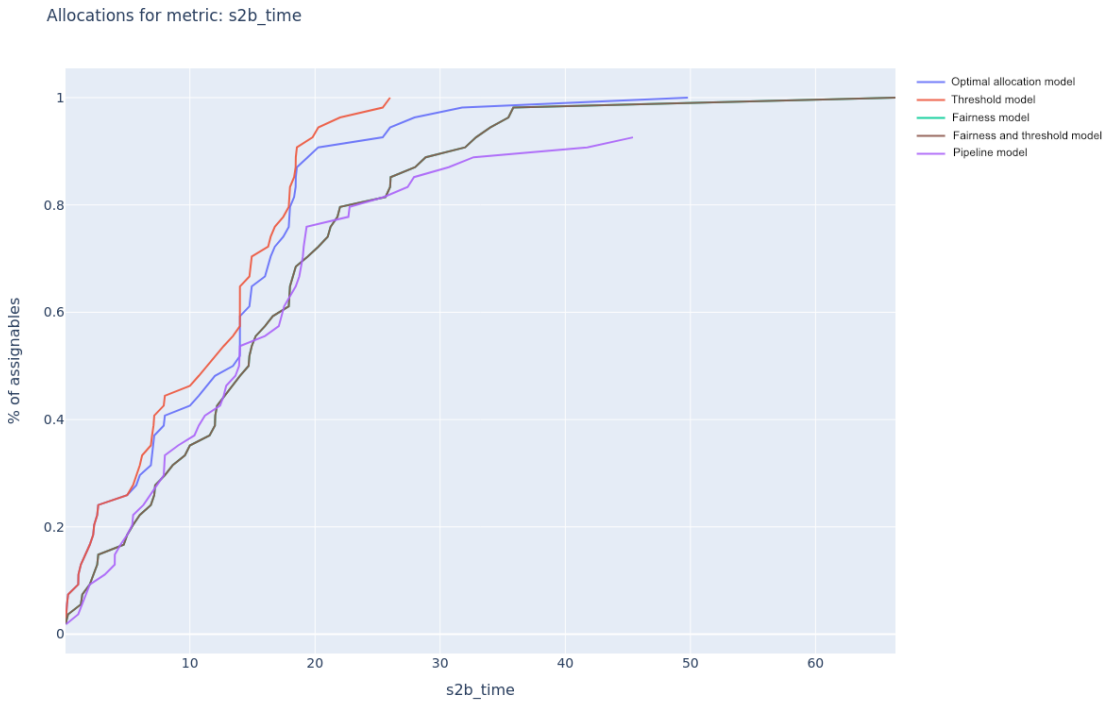


Figure 36 | Zone 4 - Noon – Store to last delivery location travel time

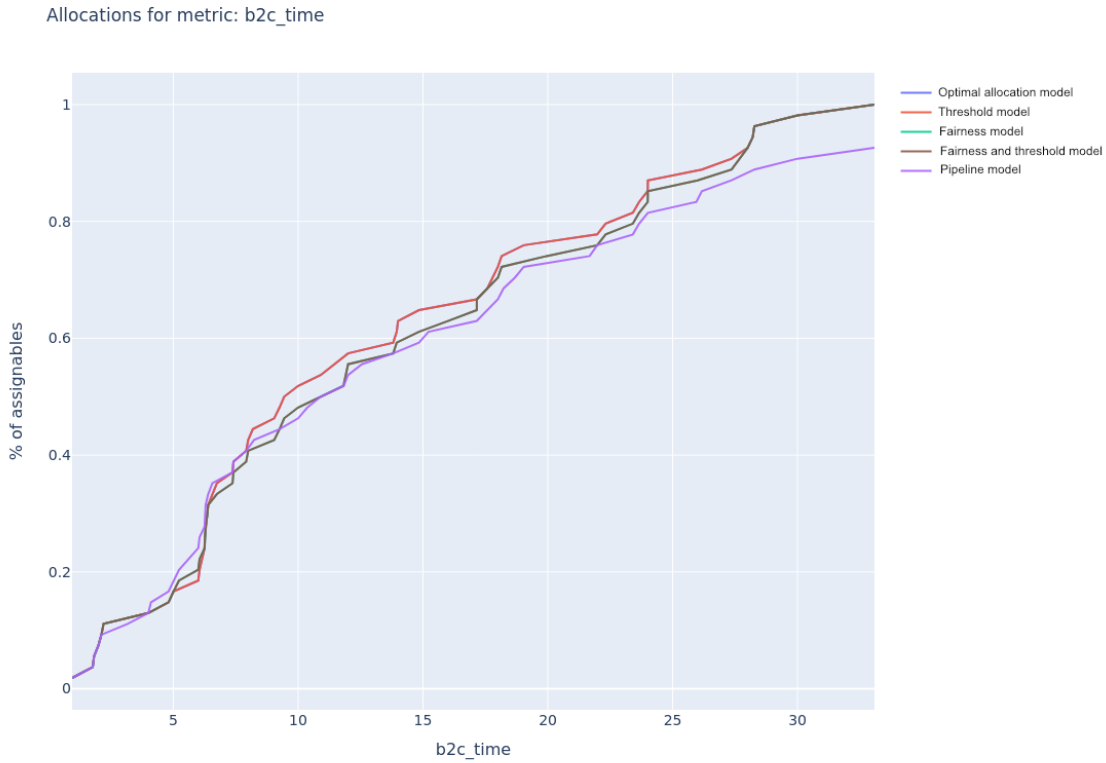


Figure 37 | Zone 4 - Noon – business preference skew performance curve

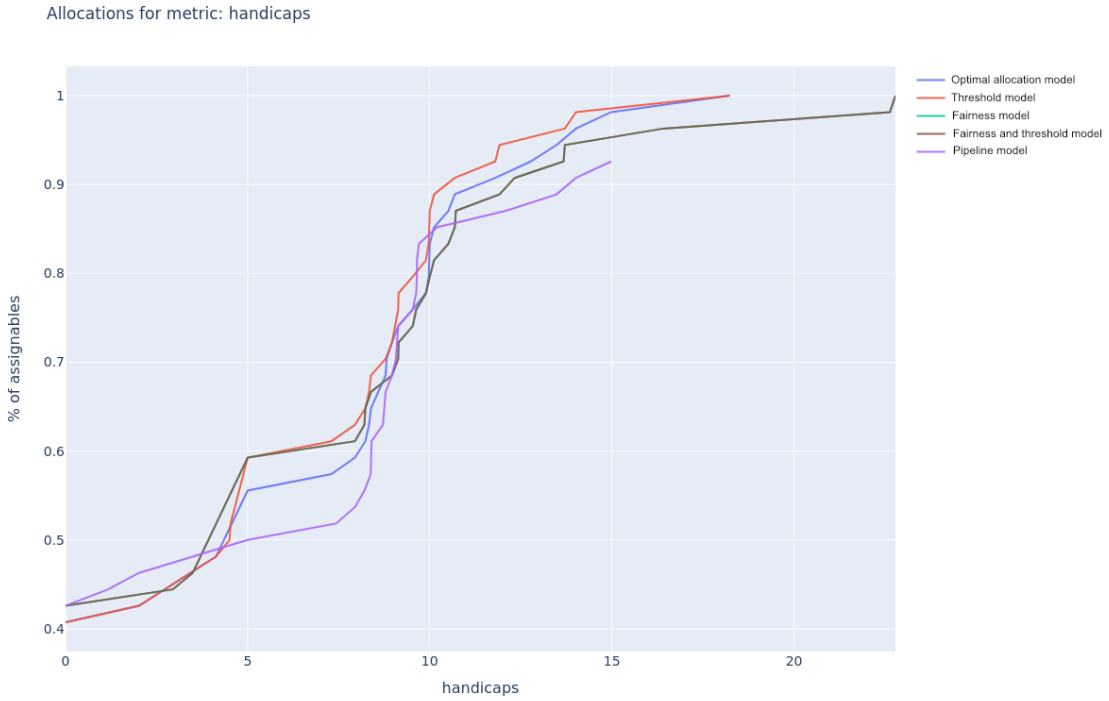
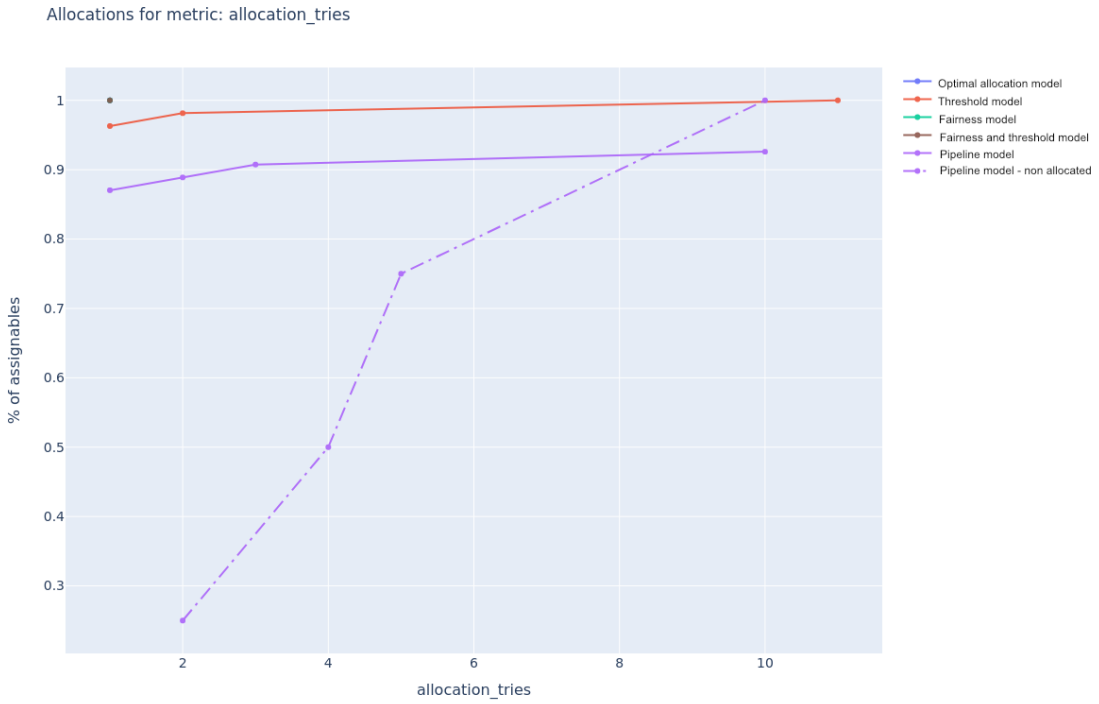


Figure 38 | Zone 4 - Noon – Attempts to find a match tries performance curve



Zone 1 – Morning run

Figure 39 | Zone 1 - Morning - Shopper to store travel time performance curve

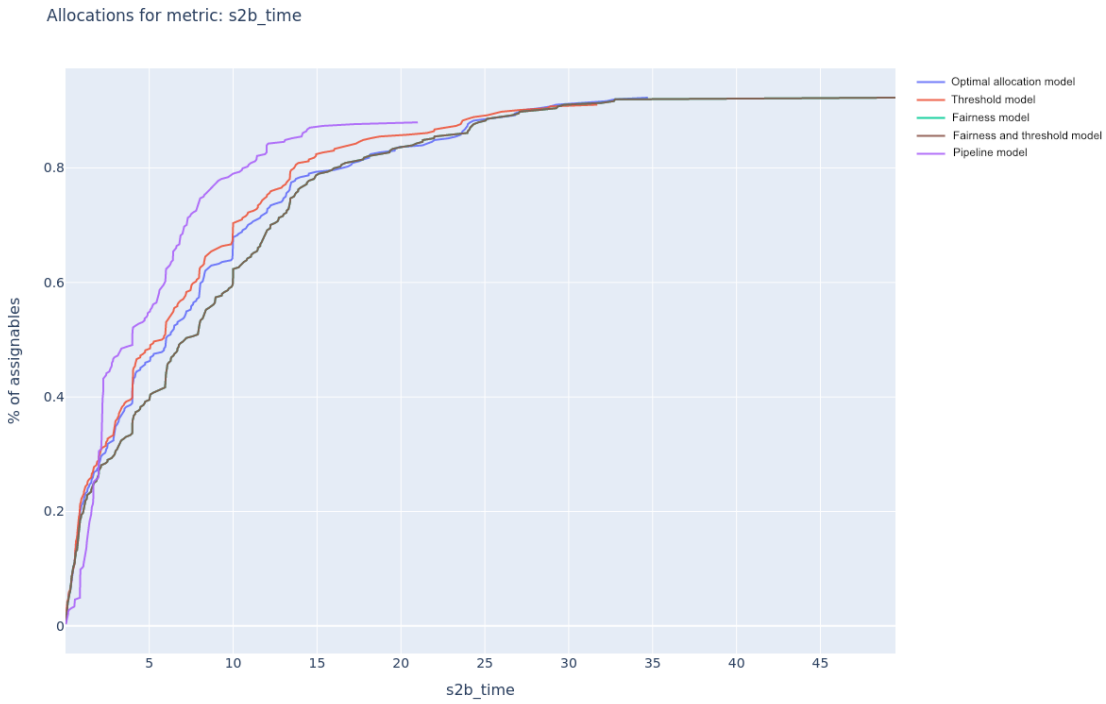


Figure 40 | Zone 1 - Morning – Store to last delivery location travel time

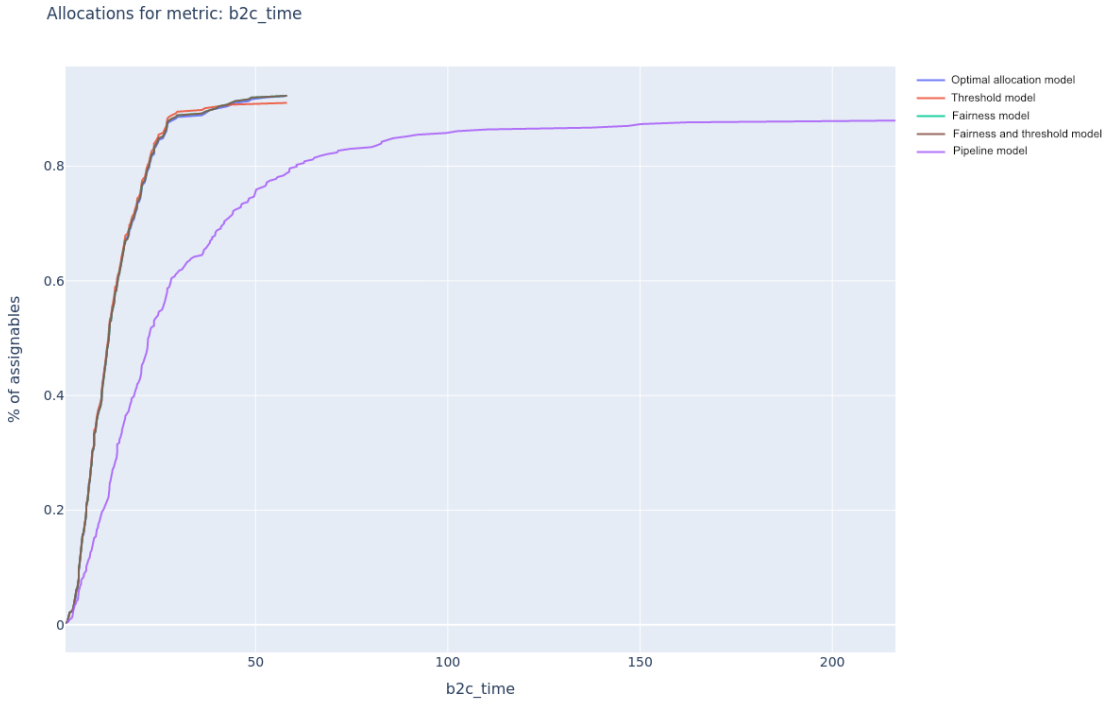


Figure 41 | Zone 1 - Morning – business preference skew performance curve

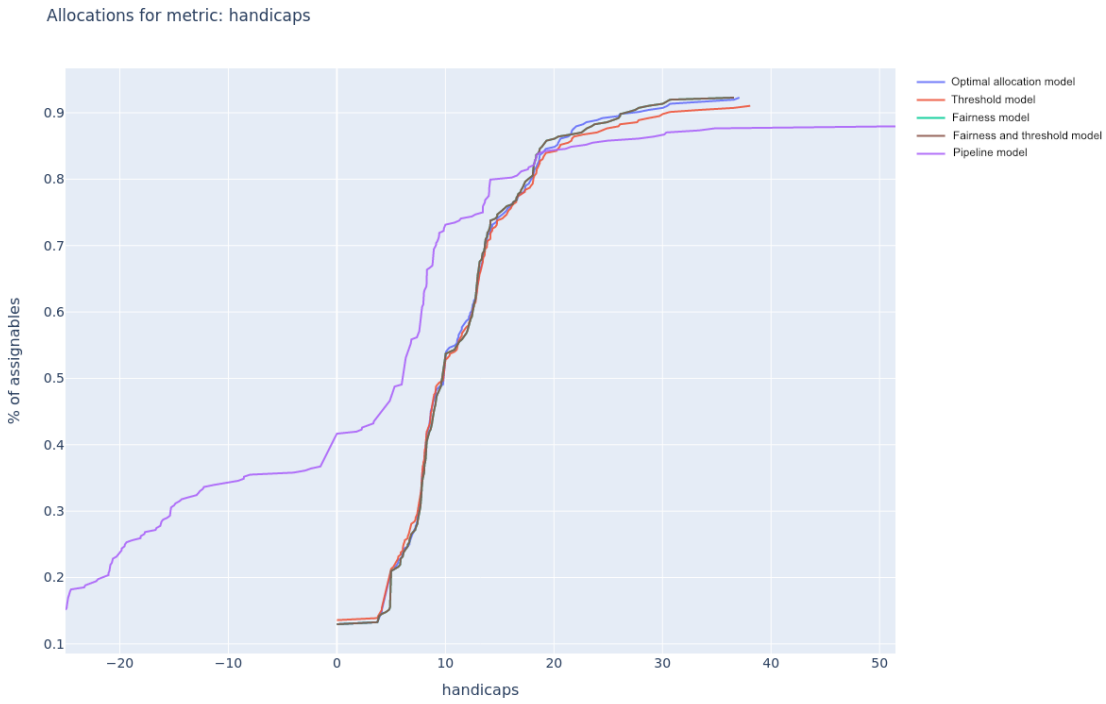
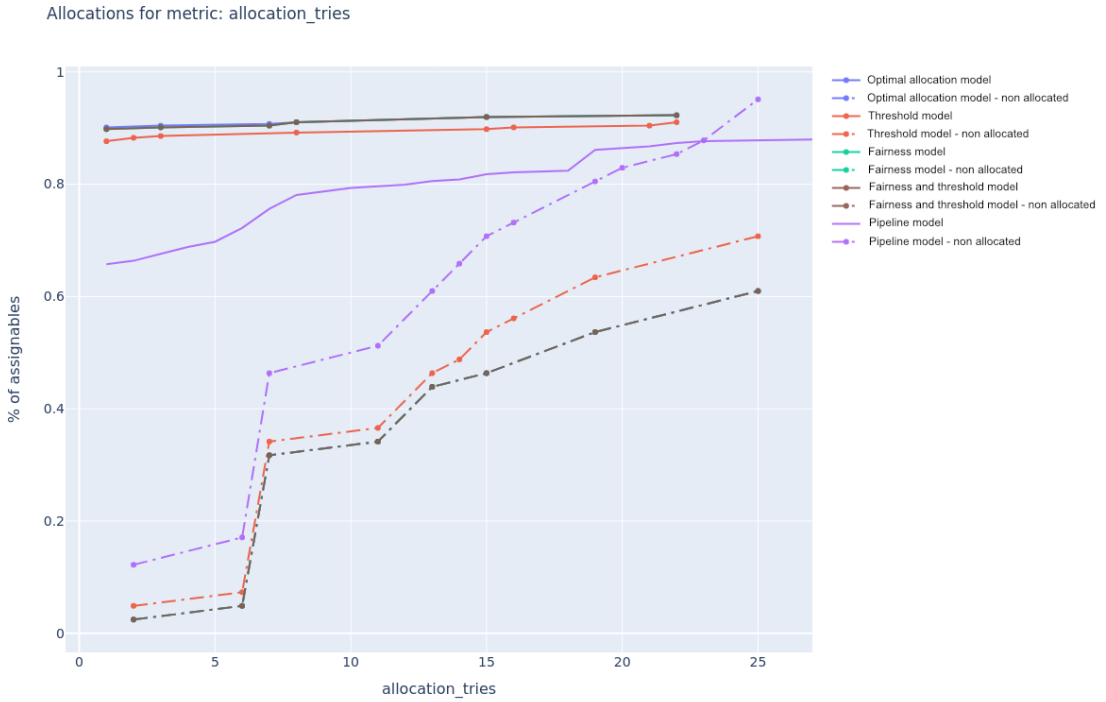


Figure 42 | Zone 1 - Morning – Attempts to find a match tries performance curve



Zone 1 – Noon run

Figure 43 | Zone 1 - Noon- Shopper to store travel time performance curve

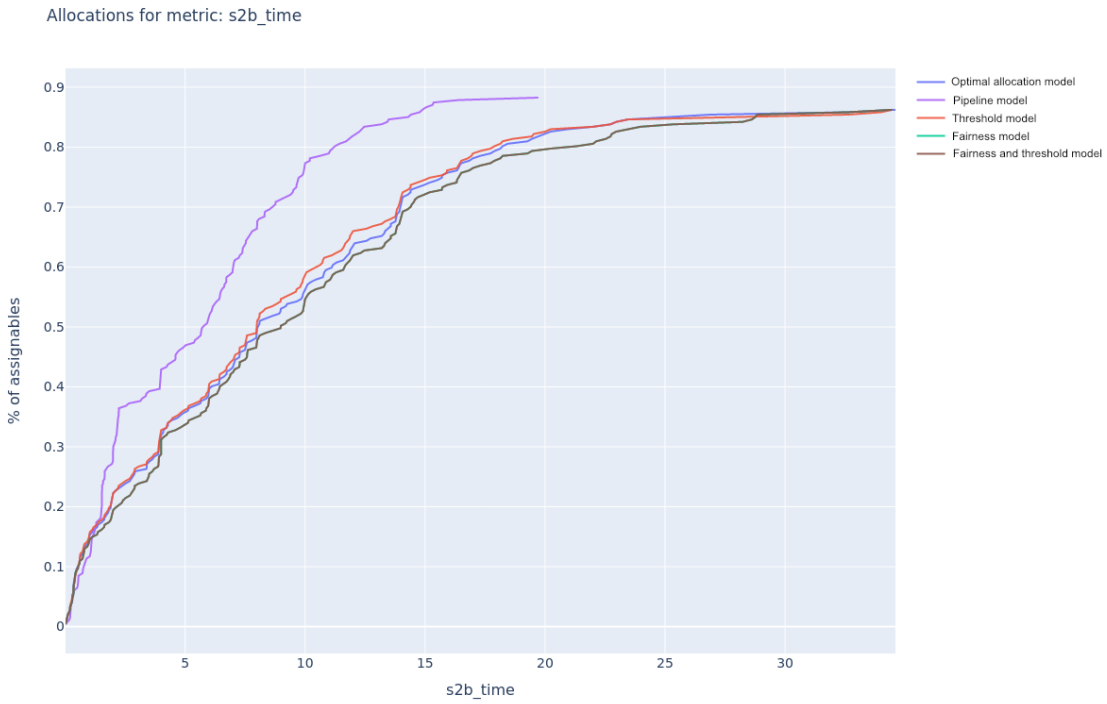


Figure 44 | Zone 1 - Noon– Store to last delivery location travel time

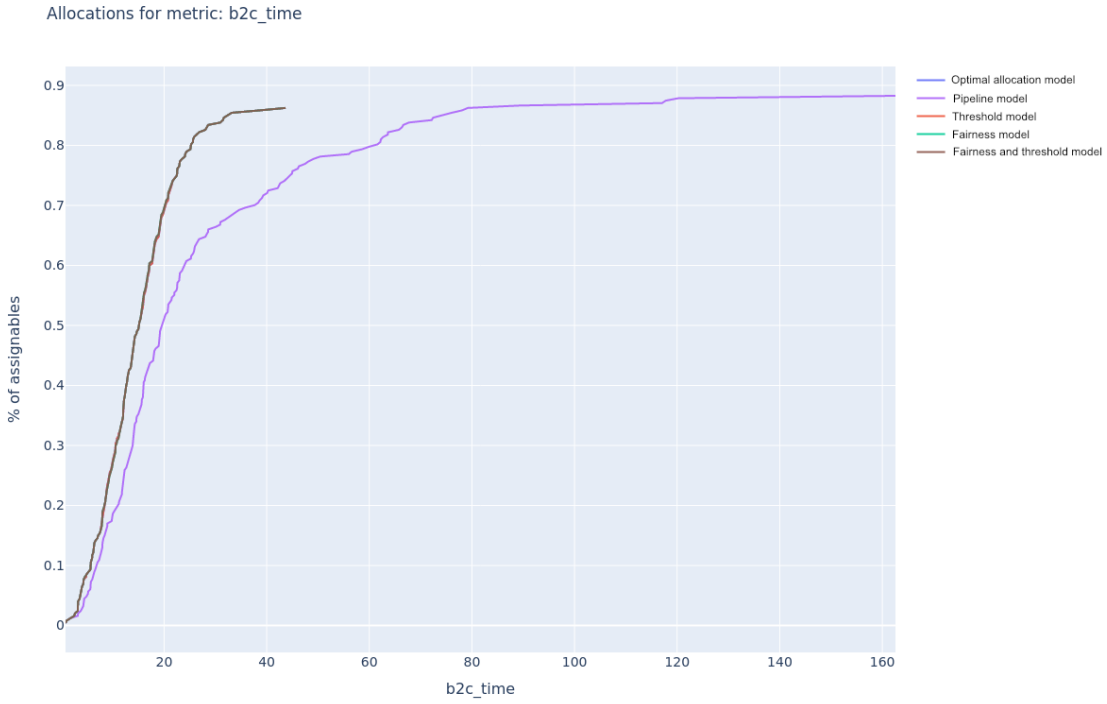


Figure 45 | Zone 1 - Morning – business preference skew performance curve

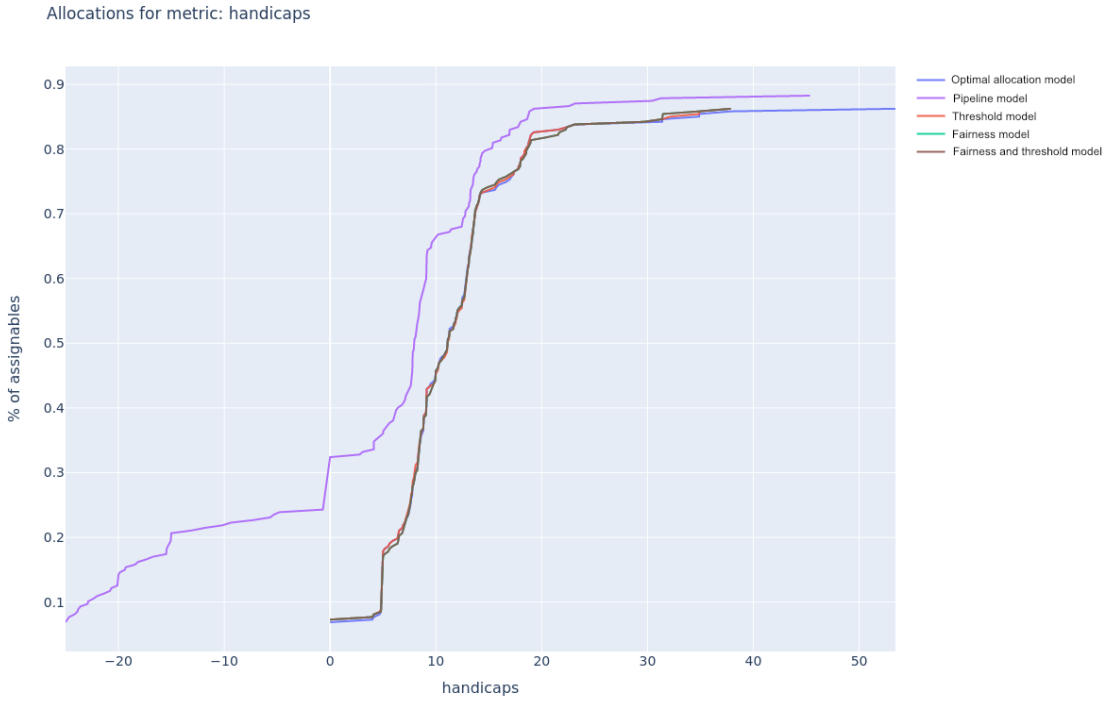
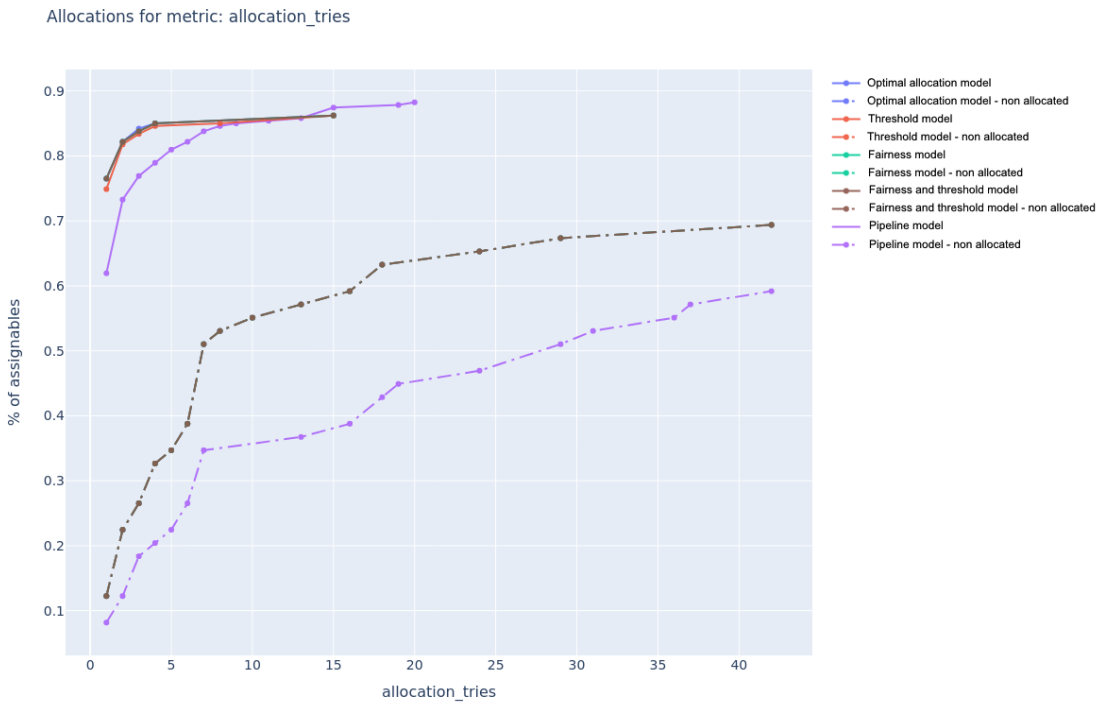


Figure 46 | Zone 1 - Morning – Attempts to find a match tries performance curve



ANNEX C.3 - CITY MODEL RUNS

Figure 47 | City - Shopper to store travel time performance curve

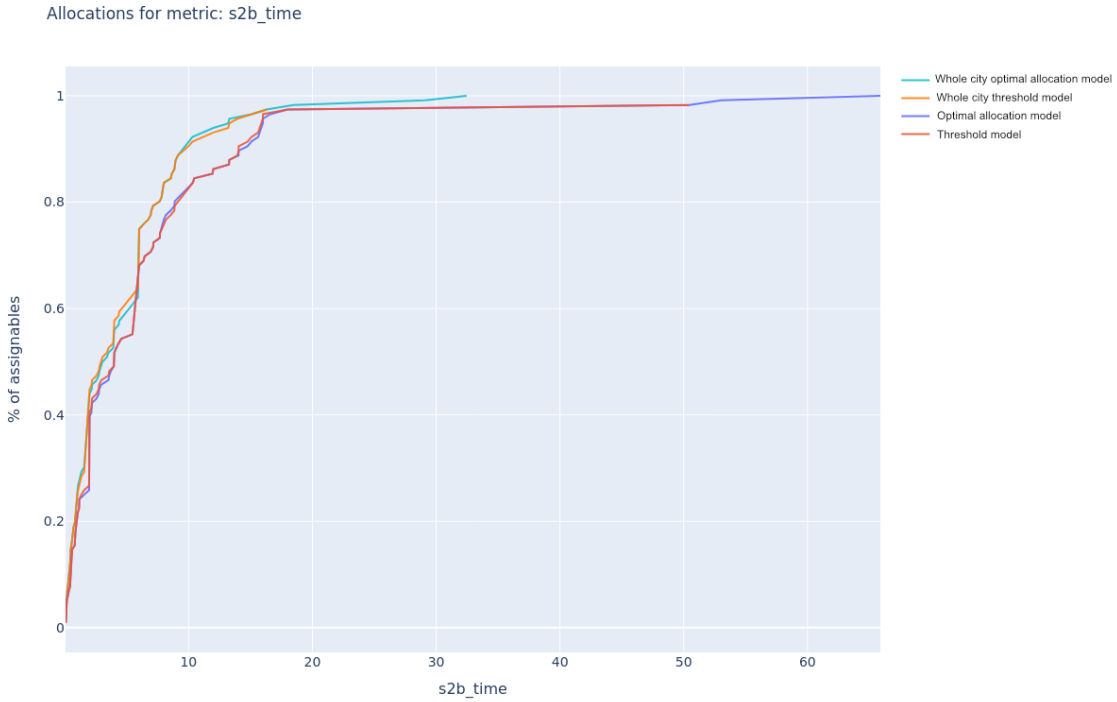


Figure 48 | City– Store to last delivery location travel time

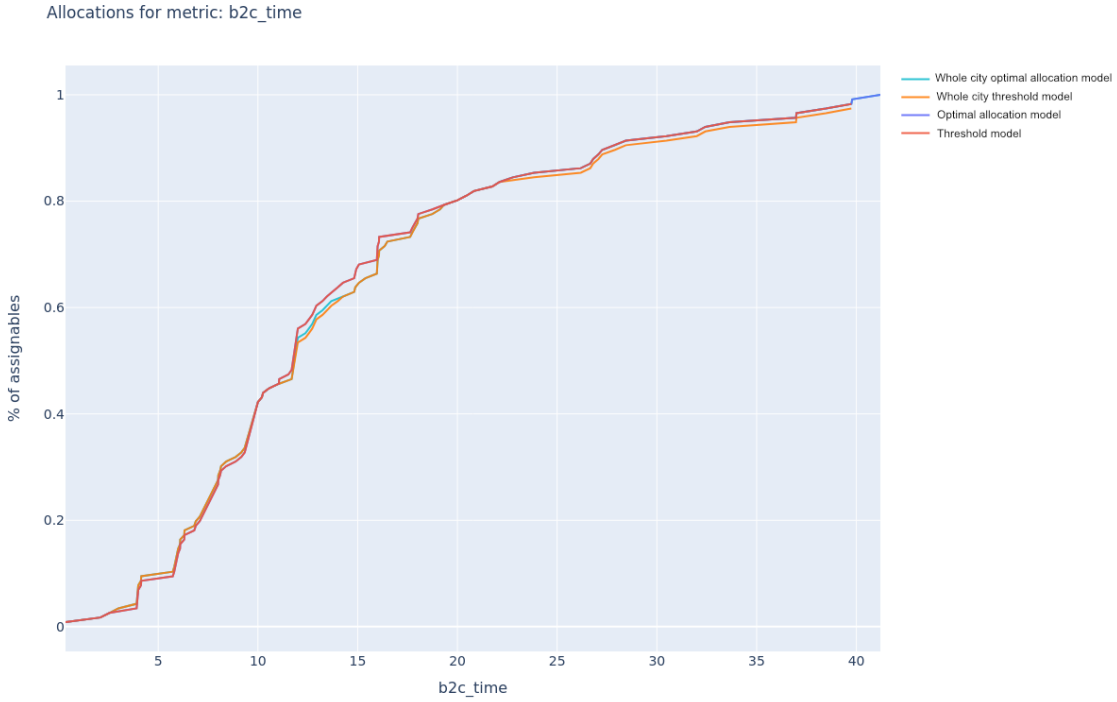


Figure 49 | City – business preference skew performance curve

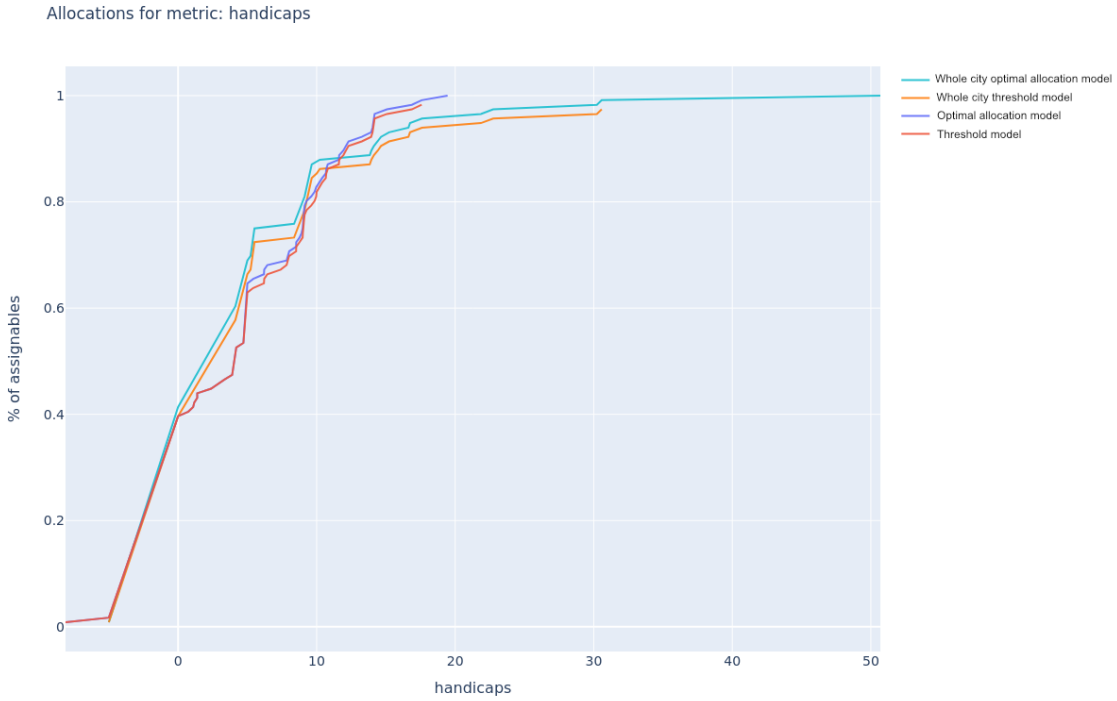


Figure 50 | City – Attempts to find a match tries performance curve