# Juggler: Multi-Stakeholder Ranking with Meta-Learning*

TIAGO CUNHA, Expedia Group, Switzerland
IOANNIS PARTALAS, Expedia Group, Switzerland
PHONG NGUYEN, Expedia Group, Switzerland

Online marketplaces must optimize recommendations with regards to multiple objectives, in order to fulfill expectations from a variety of stakeholders. This problem is typically addressed using Pareto Theory, which explores multiple objectives in a domain and identifies the objective vectors which yield the best performance. However, such approach is computationally expensive, and available commonly only through domain-specific solutions, which is not ideal for online marketplaces and their ever-changing business dynamics. We tackle these limitations by proposing a Meta-Learning framework to address the Multi-Stakeholder recommendation problem, which is able to dynamically predict the ideal settings on how business rules should be mingled into the final recommendations. The framework is designed to be generic enough to be leveraged in any item ranking domain and requires only the definition of a policy, i.e. a set of multi-objective metrics the meta-model should optimize for. The model finds the mapping between the search context and the corresponding best objective vectors. This way, the model is able to predict in real-time which is the best solution for any unforeseen search, and therefore adapt the recommendations on a search-level. We show that under this framework, the range of models one is able to build depends only on how many policies can be defined, thus offering a virtually unlimited way to address multi-objective problems. The experimental results showcase the generalization abilities of this framework and its highly predictive performance. Furthermore, the simulation results confirm the ability to approximate a policy's expectation in most cases and hints to the potential to use this framework in many other item recommendation problems.

## 1 INTRODUCTION

In recent years a surge has been observed in electronic marketplaces as a consequence of the unprecedented growth of e-Commerce which bring together sellers and buyers in a common platform. Examples of marketplaces are online travel agencies like Expedia Group or Amazon in the retail domain. Typically in these scenarios, the platforms provide recommendation systems that help to match users and suppliers. While in traditional recommendation systems one optimizes for customer utility, in marketplaces one needs to take into account multiple stakeholders. This problem, also known as Multi-Stakeholder recommendation problem [2, 28, 44], is commonly addressed through a multi-objective optimization problem, where the collection of stakeholders' goals is optimized simultaneously. An extensive collection of works is available, with examples ranging from revenue [6, 27, 31] to fairness [4, 9, 17, 28].

Despite progress in the domain, the solutions mainly focus on few and domain-dependent objectives, which makes it hard to transfer knowledge to other domains. Our framework addresses both limitations by providing a way to consider any and as many objectives as the practitioner desires. The practitioner only needs to define a meaningful policy to optimize for, meaning a collection of relevant stakeholder objectives. We show that depending on the policy, the model commonly yields predictions aligned with the policy defined, therefore approaching stakeholders' expectations. With such flexible framework, it is possible to define as many and as different policies as the business demands, leaving the practitioner with the sole responsibility of exploring policies, rather than to develop new custom-built algorithms.

To accomplish such generic framework, one needs to revisit how multi-objective problems are tackled. Typical approaches try to find the Pareto front [18, 26, 41, 45], which refers to the set of non-dominated solutions in a domain,


Presented at the MORS workshop held in conjunction with the 15th ACM Conference on Recommender Systems (RecSys), 2021, in Amsterdam, Netherlands.

i.e. objective vectors for which maximizing one objective is only possible by being detrimental to the others. Although effective under few objectives, this poses substantial problems when considering multiple: 1) when the number of objectives increases, almost all solutions become non-dominated and 2) the number of solutions required for the approximation of the Pareto front exponentially increases with the number of objectives [21]. Such constraints impose substantial restrictions when applied in real-time inference: 1) computational requirements make it extremely difficult to find tailored solutions for each problem instance, thus having to rely on sub-optimal global solutions and 2) even if the best solutions are efficiently found, the practitioner is still required to make a manual decision on which solution from the Pareto front to use [41].

To address all these issues, we argue that the ideal approach is to forego the expensive computation of the Pareto front and to focus on a set of diverse and meaningful solutions. A predictive model is then charged to decide which is the most appropriate solution, depending on the context a search occurs. Then, such prediction is used to dynamically adjust the multi-objective solution. To do so, we draw inspiration from the research area of Meta-Learning for algorithm selection [8, 39], where one learns about the learning process in order to predict the best algorithm for a new dataset [24, 36–38, 40]. Likewise, we leverage the Meta-Learning framework to predict the best objective vector amongst a set of diverse and meaningful policy-dependent solutions for any new search. Thus, in this work we make the following contributions: 1) we propose a general purpose framework Multi-Stakeholder recommendations leveraging Meta-Learning and 2) we validate the procedure in the hospitality domain using Brand Expedia data.

This document is organized as follows: Section 2 presents a summary of the relevant literature; next, Section 3 introduces the proposed framework while Section 4 shows the experimental setup used to validate it. Section 5 presents and discusses our findings and section 6 highlights our conclusions and future work.

## 2 RELATED WORK

### 2.1 Multi-Stakeholder Recommendations

We define a stakeholder in the recommendation space as any individual or group that can affect or be affected by the delivery of recommendations to users [1]. In the context of online marketplaces, they typically refer to three groups: consumers (customers/users which have a need to be met by the marketplace and which receive the recommendations to fulfill it), providers (entities which provide goods/services to the customer through the marketplace) and the system (the platform that matches consumers to providers).

In a Multi-Stakeholder recommendations, each stakeholder has a different expectation that the platform has to meet, or at least to satisfy; that is a scenario which is more realistic in the real world that the basic task to optimize for customer utility in academic research [22]. Since such problem is not trivial, much research has been devoted to this topic [28, 44]. More recently, there was a surge of proposed approaches in industrial settings, mainly due to e-Commerce growth: balancing semantic match and job-seeking intent features [35], to learn how to balance hotel relevance and compensation [31] and to balance customer relevance and advertising revenue in music platforms [3, 4].

The Multi-Stakeholder problem can be formulated through the multi-objective optimization setup, where all stakeholder's goals are considered as one single super-set of objectives to optimize for. Formally, the problem is defined as [18]: given $m > 1$ objective functions $\theta_1 : X \rightarrow \mathbb{R}, \ldots, \theta_m : X \rightarrow \mathbb{R}$ which map a decision space $X$ into $\mathbb{R}$, a multi-objective problem is defined as by:

$$\min \theta_1(x), \ldots, \min \theta_m(x), x \in X \tag{1}$$

Traditionally, the multi-objective problem is tackled with Pareto Theory methods, which aims to find the Pareto front, i.e. the set of non-dominated vectors for any particular problem. Such set describes all vectors for which it is only possible to perform better in one dimension, if it is accompanied by a decrease in performance in another. Such front is ideal to uncover the relationship between objectives and, consequently, the best solution. The quintessential notion in Pareto is the dominance, i.e. how to compare two instances to make sure one is better than the other. It is defined as [18]: given two vectors in the objective space $y^{(1)} \in \mathbb{R}^m$ and $y^{(2)} \in \mathbb{R}^m$, then the point $y^{(1)}$ Pareto dominates $y^{(2)}$ if and only if: $\forall i \in \{1, \ldots, m\} : y_i^{(1)} \leq y_i^{(2)}, \exists j \in \{1, \ldots, m\} : y_j^{(1)} < y_j^{(2)}$. More plainly, it states the first vector is not worse in each of the objectives than the second vector and better in at least one [41].

The typical approach to solve the multi-objective problem consists of two steps: to simplify the objective function and to apply an optimization algorithm to find the solution. To simplify the problem, practitioners use scalarization functions, which are mathematical constructs used to convert multiple objective functions into a single one. This way, the problem can be solved using standard optimization algorithms [18]. Then, to find the solution, practitioners usually employ evolutionary algorithms, as these are capable of finding well-distributed solutions along the Pareto front [18]. Such algorithms take advantage of natural evolution concepts of selection, mutation and combination to explore the objective space until a near-optimal solution is found. For more details, see [26, 41, 45].

### 2.2 Meta-learning

The No Free Lunch Theorem [43] refutes the existence of a so-called super-algorithm: a single best, universal learning algorithm able to obtain the best possible performance for every instance of a given task [39]. The justification lies in the observation that if all possible data distributions are equally likely, any pair of learning algorithms will, on average, have the same performance. Therefore, for any algorithm, superior performance over one group of task instances is compensated with inferior performance over another.

Researchers have then decided to focus on understanding each algorithm's behavior/bias in order to ascertain when they will be most successful. Meta-Learning is one of the existing tools to tackle the problem, which focuses on using Machine Learning to understand Machine Learning algorithms (and their configurations) in order to improve their results in future applications [32, 36]. These are commonly addressed by learning meta-models which find the relationships between data characteristics and learning performance [36, 38]. This algorithm selection problem has first been formalized by Rice [34], and it states that given:

- the problem space $\mathscr{P}$, representing the set of problem instances for which predictions will be made;
- the feature space $\mathscr{F}$ containing measurable characteristics for each instance of $\mathscr{P}$;
- the algorithm space $\mathscr{A}$ as the set of all available algorithms for solving the problem;
- the performance space $\mathscr{Y}$ that shows the mapping from algorithms to performance metrics,

the problem can be stated as: for a given problem instance $x \in \mathscr{P}$, with features $f(x) \in \mathscr{F}$, find the selection mapping $M(f(x))$ into the algorithm space $\mathscr{A}$, such that the selected algorithm $\alpha \in \mathscr{A}$ maximizes the performance mapping $y(\alpha(x)) \in \mathscr{Y}$ [37].

Using Meta-Learning and Recommender Systems simultaneously is not a new topic: there are works that focus on the selection of the best recommendation algorithm for a new dataset [5, 12–14, 16], selection of the best algorithm for each user [10, 11, 33] and even how to use Recommender Systems to tackle algorithm selection tasks [15, 30]. However, to the best of our knowledge, this is the first documented solution to tackle the Multi-Stakeholder recommendation problem using Metalearning.

## 3 JUGGLER

We focus on the problem of item ranking in online marketplaces, namely how to rank multiple items given the customer's query or based on the customer profile. In this setting, a ranking $\pi$ is defined by decreasingly sorting items $i \in I$ by their score $s(i)$.

In this work, we define our objective function as a sum of all stakeholder's objectives, by building on the previous work [31]. The score $s$ for any item $i$ in this context is given as:

$$s(i) = \sum_{k=1}^{K} a_k(i) \tag{2}$$

where $a$ refers to each of $K$ adjustments, i.e. functions which score each item with regards to some specific objective(s). Usually these refer to mathematical formulas to enforce business rules, but can be more complex: for instance, the predictions from other Machine Learning models. Thus, this formulation is flexible enough to allow any number and nature of adjustments, which is advantageous in ever-changing online platforms.

This formulation offers a straightforward way to address the multi-objective optimization problem, namely through a linear scalarization [18], i.e. by introducing an individual weight per adjustment. This is achieved by modifying Equation 2 accordingly:

$$ws(i) = \sum_{k=1}^{K} w_k \times a_k(i) \tag{3}$$

where each parameter $w$ identifying a specific adjustment's weight. From such definition, it is possible to define a ranking of items, simply by sorting the items for a given user in a decreasing order. Thus, the ranking $\pi$ over a subset of items $p \subseteq I$ referring to search is the sequence of items given by: $\pi = (i_1, \ldots, i_n), ws(i_1) \geq ws(i_n), \forall i \in s$. Now, inspired by Equation 1, we can state our objective function to optimize the weighted item scores in a way that all objectives are maximized:

$$\max \theta_1(\pi), \ldots, \max \theta_m(\pi) \tag{4}$$

### 3.1 Framework overview

The main hypothesis in this paper is that Multi-Stakeholder recommendations can be addressed through Meta-Learning. We take inspiration from algorithm selection task and adapt it to our problem, by considering problem instances $x$ as individual searches $p$ and replacing algorithm $\alpha$ by objective vector $w_k$. This parallelism will become evident in Section 3.3, when we explain how we convert objective vectors into a discrete set of meta-labels.

This way, the Meta-learning problem addressed is: for a given search $p \in \mathscr{P}$, with features $\mathscr{f}(s) \in \mathscr{F}$, the meta-model aims to find the selection mapping $M(f(s)) \rightarrow \mathscr{A}$, such that the selected objective vector $w_k \in \mathscr{A}$ maximizes the performance mapping $y(w_k(s)) \in \mathscr{Y}$. Figure 1 presents an overview of the training and inference workflows in the proposed Juggler framework, in order to illustrate how it can be leveraged in online marketplaces.

In training, we leverage historical searches to build the meta-examples, i.e. a set of points in the meta-feature space, with an associated label. Each point refers to a search and it is defined by the meta-feature extraction process; the labels are constructed via simulations, by assessing which is the ideal multi-objective vector calibration for each search.

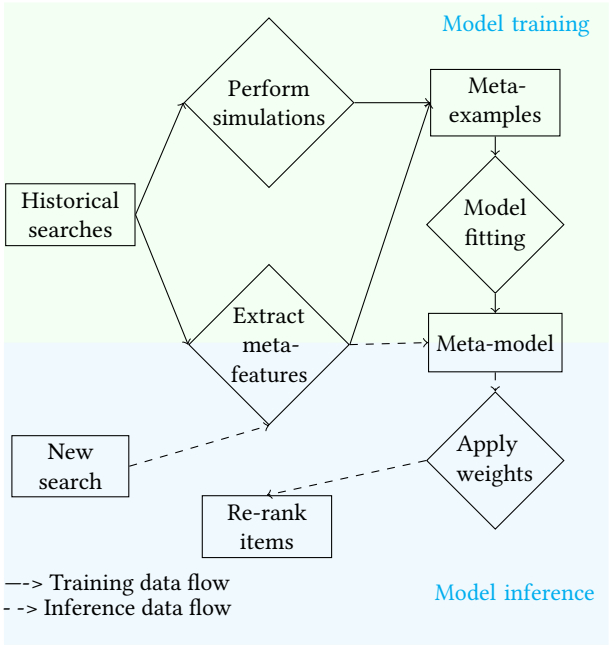


Fig. 1. Juggler framework.

With such meta-examples, the meta-model is fitted using a standard Machine Learning procedure. When placed into production, it is ready to perform real-time inference.

The inference step involves then simply submitting a new search to the same meta-feature extraction process used in training, in order to be able to place the new search in the meta-feature space. Then, the meta-model predicts the best multi-objective vector for this particular search, based on patterns found in the simulations employed in training. The prediction is used to re-rank the items, following the predefined scoring system.

We shall discuss in the remainder of this section how do we perform simulations (by introducing the concepts of policy and how these are used to define meta-labels) and how do we characterize searches through meta-features.

### 3.2 Meta-performance: defining policies

We will describe now the set of metrics $\mathscr{Y}$ that we want to optimize with Meta-Learning. As established in the Multi-Stakeholder literature, there are many objectives one can optimize for at any given time: it is then a question to define which matter most in a specific use case. Common solutions focus on few objectives, highly dependent on the problem at hand. We address this limitation by defining the concept of policy.

A policy $\Gamma$ refers to a set of multi-objective metrics. Such metrics are already naturally defined in each multi-objective function $\theta_m$ in Equation 4, i.e. $\Gamma = \{\theta_1, \ldots, \theta_m\}$. In our setup there is no constraint over which metrics nor how many metrics one chooses to optimize for, as long as a valid sets of metrics is provided. Following theoretical considerations on ranking metrics [42], we consider a metric to be valid if it possesses the distinguishability property, i.e. it can distinguish two different rankings. We require only its output to be a continuous scalar value, with higher value meaning better.

Many widely-accepted ranking metrics exists today follow such constraints: for instance, Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision and Mean Reciprocal Rank [20]. These, however, are not able to capture all the dynamics of the multi-objective recommendation problem, for instance in terms of fairness or how well aligned the outcome is with a specific business objective. To address this gap, we propose 2 different metric classes for rankings to be in combination and separately in our framework: Ranking Correlations and Ranking Fairness. The goal is to provide a skeleton to enable to optimize for objectives for which there are no known metrics yet.

*Ranking Correlations.* This metric class offers a way to understand how well does a specific ranking perform on a given attribute, when compared against the ideal setting. For instance, how correlated a given search is with the increasing price-aware sorting of the same items tells us how close we are to a ranking with cheaper properties on the top positions. Similarly, such operation can be used for any item attribute which enables to derive a ranking of items. Since we shall operate under a ranking setup, we take advantage of Kendall's *tau* ranking correlation to measure correlations to the ideal scenario. Formally, the correlation between a ranking $\pi$ and the ideal ranking $\pi^*$, given by a ranking function over a particular attribute, is calculated by first assessing how many concordant $n_c$ and discordant $n_d$ pairs there are when comparing pairwise items in both rankings. Kendall's $\tau$ is thus given by:

$$\tau = \frac{n_c - n_d}{n(n-1)/2} \tag{5}$$

*Ranking Fairness.* One type of objectives that has recently received substantial attention in the literature is the fairness and diversity [2, 9, 17, 19]. Such approaches try to ensure the distribution of predicted items per class is respected regardless of the context on which the prediction occurs. For instance, we could optimize for a ranking to show all existing property types in the destination region, while trying not to hurt relevance of predictions. To that effect, we build on a Group Fairness metric [29], originally designed to ensure fairness of item popularity, and generalize it to ensure fairness of any item attribute. Considering an attribute which can be divided into $K$ classes, with each group being represented by $I_j, j \in \{1, \ldots, K\}$. Thus, one can calculate the Group Fairness metric by considering the frequency of items in each class:

$$GF(s) = \sum_{j=1}^{K} \sqrt{|i|}, \forall i \in I_j \tag{6}$$

With such metric classes, one can define a variety of metrics, which in turn means it is trivial to define multiple policies. One key advantage of this formulation is that multiple meta-models can be inferred simply by switching the policy. This is particularly noteworthy since it means any policy can be used, even though it might at times lead to sub-optimal performance: it is up to to practitioner to decide which are the meaningful objectives to optimize for and to confirm through simulations and AB testing whether the expected outcome is met. Alternatively, such policy could be defined by product or analytics teams, based on domain knowledge and business dynamics.

### 3.3 Meta-labels: choosing weights

To find the objective vectors $w_k$ to be used as meta-labels is no trivial task and depends heavily on the policy $\Gamma$: it is unlikely that two policies should share exactly the same objective vectors. To that end, we define a procedure which aims to explore a constrained objective space in order to select a few good candidates. These candidates constitute all

available options the model has to dynamically weight the adjustments at inference time, thus reducing the problem from finding ideal weights to predict the best candidate. The procedure to find the meta-labels follows these steps:

(1) **To constrain each objective into a range**: each adjustment $a_k$ must be constrained within a lower and higher boundary, i.e. $[\min(w_k), \max(w_k)]$. The boundaries should be defined in such a way that the adjustment is allowed to only vary so much from the original value (i.e. $w_k = 1$) - otherwise, it may lead to unforeseen negative effects on the overall ranking. Such range should be tuned through experimentation and/or domain knowledge.

(2) **To partition the objective space into sections**: instead of exploring all possible combinations in the feasible region, we focus on the most interesting sections: the extreme ends of the bounded objective space (to explore the objective space in search of meaningful changes in ranking) or closer to the default values (to find the best fit for the current default setup). Thus, each space is defined as a polygon, covering 1/3 of each dimension, i.e. $(\max(w_k) - \min(w_k))/3$. The outer polygons start from the extreme ends of objective range, while the balanced section is centered on the origin point. As an example of the partition of an objective space for 2 objectives is shown in Figure 2. Notice this procedure generates 5 meta-labels when using 2 objectives, which is a good trade-off for the classification algorithm we aim to employ.

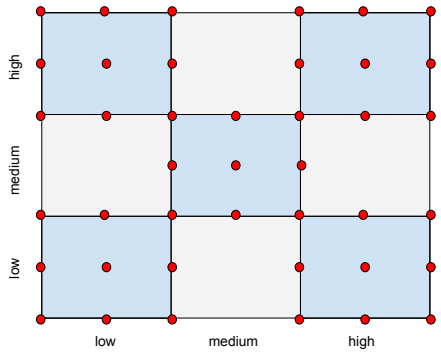


Fig. 2. Meta-label sections in an objective space for 2 objective functions. Notice it identifies 5 sections (shaded in blue), which will translate in the same number of meta-labels.

(3) **To discretise the space into representative candidates**: we sample some examples from each section to serve as candidates as, once again, there is not necessarily any change in ranking by considering 2 contiguous objective vectors. To do so, we employ a grid-search approach, similarly to another procedure in the literature [35].

(4) **To create solutions using candidates**: a solution $c$ refers to a set of objective vectors, containing exactly one candidate per section, i.e. $c = \{w_k^1, \ldots, w_k^5\}$ with $w_k^N$ referring to one selected objective vectors in each of the 5 available sections. The set of all solutions $C$ refers to all combinations of candidates across sections.

(5) **To evaluate solutions**: all solutions are evaluated on policy $\Gamma$ in all searches. First, each objective vector in each solution is submitted to Equation 3 in order to score and, afterwards, rank each item within a search. Then, each candidate is evaluated individually in each objective $\theta_m$ with regards to each search, which enables to calculate its average ranking across all objectives.

(6) **To identify the best solution per policy**: the best solution is simply the one which achieves the lowest average rank across all objectives. In the presence of ties, the candidate closer to the default weights is selected, since there is no gain in deviating from the default behaviour. Algorithm 1 summarizes the evaluation procedure explained in the previous two points.

**Algorithm 1** Evaluate and select meta-labels

**Require:** set of searches $\mathscr{P}$, set of solutions $C$, policy $\Gamma$

**for** $c \in C$ **do**
  **for** $p \in \mathscr{P}$ **do**
    **for** $w_k^l \in c$ **do**
      **for** $i \in p$ **do**
        $ws(i) \leftarrow \sum_{k=1}^{K} w_k^l \times a_k(i)$ ▷ Score items
        $\pi \leftarrow rank(ws(i))$ ▷ Rank items
      **for** $\theta_m \in \Gamma$ **do**
        $\mathscr{Y}_{\theta_m}(p, w_k^l) \leftarrow \theta_m(\pi)$ ▷ Evaluate candidate
    $\mathscr{Y}(c,p) \leftarrow \frac{\sum_{m=1}^{|\Gamma|} \mathscr{Y}_{\theta_m}(p, w_k^l)}{|\Gamma|}$ ▷ Score solution in search
$\mathscr{Y}(C) \leftarrow \frac{\sum_{1}^{|\mathscr{P}|} \mathscr{Y}(c,p)}{|\mathscr{P}|}$ ▷ Evaluate over all searches
return $\arg\min rank(\mathscr{Y}(C))$ ▷ Select best solution

### 3.4 Meta-features: characterizing context

One must now define the feature space $\mathscr{F}$ which will allow to describe the context on which each search occurs. Meta-Learning literature presents several approaches to describe a problem, but as far as we are aware, there are no standard solutions to describe rankings. We propose two meta-feature classes, similarly to what was done for metrics in Section 3.2, which can be used on any ranking: Ranking histograms and Ranking comparisons.

*Ranking histograms.* To take advantage of histograms in Meta-Learning is not new [23] and it is appreciated since they are capable of summarizing a given feature $f$ using a single summary function, i.e. the ratio of elements by bucket. It requires, however, a significant amount of data in order to extract meaningful patterns - to do so, we focus on a specific ranking threshold $\psi$ to extract data from, large enough to be meaningful. Thus,

$$\mathscr{f}_{RH} = \textsc{Histogram}(\{f(i)\}), \forall f \in F, \forall i \in s : rank(i) < \psi \tag{7}$$

*Ranking comparisons.* Here, we shift the attention from the item features to item scores: we achieve this by comparing the same item set in a search using different scoring mechanisms, i.e. using subsets of the adjustments from Equations 2 and 3. Thus, considering two rankings $\pi_1$ and $\pi_2$ of the same items, constructed with different scoring functions, we employ multiple functions ranking comparison functions to extract summary statistics of their similarity. Examples of such metrics include correlation metrics such as Kendall's $\tau$ (see equation 5), Cosine similarity, Pearson Correlation, etc. Formally, for a set of ranking correlation functions $\Omega$, we have:

$$\mathscr{f}_{RC} = \{\omega(\pi_1, \pi_2)\}, \forall \omega \in \Omega \tag{8}$$

The full set of meta-features is then $\mathscr{F} = \mathscr{f}_{RH} \cup \mathscr{f}_{RC}$, which may translate into a set of hundred or even thousands of meta-features, depending on how many different parameters are used to instantiate the meta-feature classes. To tackle this issue, the procedure assumes the application of a feature selection process, which shall find the best set of meta-features per policy. We make no assumptions on the feature selection procedure, but focus particularly on correlation feature selection.

## 4 EXPERIMENTAL STUDY

We validate Juggler in the context of top-N property recommendations in Expedia brand. We use both private data and proprietary recommendation algorithms, which have been designed and are continuously tuned to address Multi-Stakeholder recommendations. In this context, each hotel $h$ is scored using the following rule:

$$s(h) = \text{UTILITY}(h) + \text{COMPENSATION}(h) + \dots \tag{9}$$

where UTILITY refers to the relevance each item holds to meet the customer's needs and COMPENSATION refers to an adjustment responsible to penalize lower margin properties [31]. Notice the formula is incomplete, meaning there are many hidden adjustments in place to account for many other marketplace objectives. We account for all adjustments in the results presented in this document, which makes the problem both more realistic and more difficult to solve. Notice, for instance, that otherwise we could not use the same weight on both adjustments, as it would always yield the same ranking outcome, rending some combinations worthless.

### 4.1 Data and meta-features

The data used for meta-model fitting refers to a sample of 6 million searches from 2019. The data is split into training/tuning/validation subsets by randomly sampling searches according to the 70/10/20 rule, respectively.

Each search is a collection of properties, and it has both search-specific attributes and property-specific attributes. The search is also accompanied by the utility scores per property and has all features required to compute the remaining adjustments. The complete list of attributes can be organized as follows:

- Search features: id, destination (identifier, country and type), check-in/out derived signals (day of week, week of year, month), booking window and length of stay. These require no pre-processing, thus becoming meta-features by default.
- Property features: price, margin, star rating, guest rating, distance to center, production statistics, etc. We use these attributes as features $f$ in $\not{f}_{RH}$ meta-features, with $\psi = 30$. No other thresholds have been used for simplicity.
- Property scores: UTILITY, COMPENSATION and all its variations are calculated for all properties. Then, the following $\omega$ functions will be used to compare rankings: Kendall's $\tau$, Pearson's correlation and Cosine similarity.

Afterwards, models are simulated using a more recent data sample with 100000 searches, which covers the period of January 2020 until March 2021. The features are exactly the same as before.

### 4.2 Policies

Following the convention from Equation 3, we study the Multi-Stakeholder problem through the following rule:

$$ws(h) = w_u \times \text{UTILITY}(h) + w_c \times \text{COMPENSATION}(h) + \dots \tag{10}$$

defining variables $w_u$ and $w_c$, responsible for weighting UTILITY and COMPENSATION scores, respectively.

We take advantage of NDCG as a measure of customer utility, given it is a standard choice in Recommender Systems. Furthermore, we explore Ranking Correlations and Ranking Fairness through Kendall's $\tau$ and Group Fairness metrics, presented in Equations 5 and 6. These will be used to further explore the rankings produced during simulations, in an attempt to understand how well do we perform in specific business objectives.

In Ranking Correlations, we focus on specific attributes to create the ideal rankings: lowest price, highest margin, lowest distance, highest guest rating and highest number of reviews. In order to ensure all objectives are optimized, the dimensions which aim for lower values - price and distance - are ranked in increasing order. For simplicity, we refer to these as $\tau_{price}$, $\tau_{margin}$, $\tau_{distance}$, $\tau_{rating}$, $\tau_{reviews}$, respectively.

Group Fairness focus on specific marketplace components we wish to improve upon by offering a more balanced solution of each category, namely: markets within destination, property types, property recency in marketplace (in years), property production statistics and property branding versus non branded properties. These will be denoted as $GF_{market}$, $GF_{type}$, $GF_{recency}$, $GF_{prod}$, $GF_{brand}$ henceforth.

The objectives can thus assigned to the following stakeholders:

- Customers: associated with conversion and property quality metrics such as NDCG, $\tau_{price}$, $\tau_{distance}$, $\tau_{rating}$, $\tau_{reviews}$
- Providers: which aim mostly to have a fair exposure rate, regardless of their characteristics, are represented by all Group Fairness metrics, i.e. $GF_{market}$, $GF_{type}$, $GF_{recency}$, $GF_{prod}$ and $GF_{brand}$.
- System: which focus mainly in conversion and revenue - namely through NDCG and $\tau_{margin}$ - although it extends its interest also to Group Fairness metrics to maintain marketplace health.

To make sure to design policies that are able to address multi-objective problems in a meaningful way, it is important to focus on competing objectives, ideally addressing all stakeholder's objectives. One way to ascertain such competition is to ensure low correlation between objectives. The correlation matrix for all objectives proposed is shown in Figure 3. With the exception of Group Fairness objectives, all other candidates have low correlation amongst themselves.

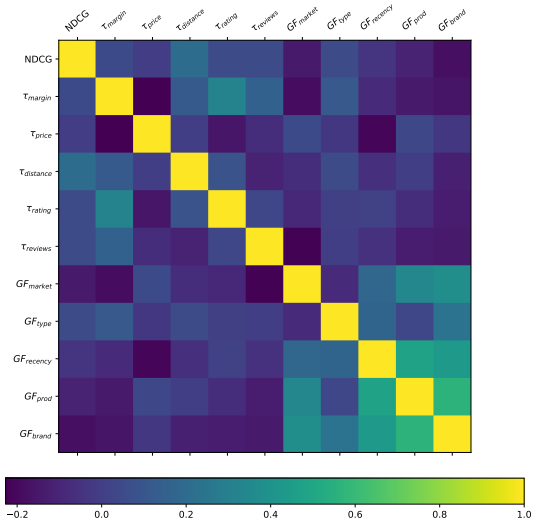


Fig. 3. Correlation matrix for all objectives.

Under this context, we explore the following policies:

- Policy I: focus on Customer and System objectives only: NDCG, $\tau_{price}$, $\tau_{margin}$, $\tau_{distance}$, $\tau_{reviews}$, $\tau_{rating}$. The rationale for this policy is to understand what happens if we disregard Producers altogether.

- Policy II: aims to find the best ranking in terms of customer relevance and marketplace fairness, thus tending to customer and partner's interests, i.e. NDCG, $GF_{market}$, $GF_{type}$, $GF_{recency}$, $GF_{prod}$, $GF_{brand}$. In this case, we forego the System's own interests, attempting to straighten the relationship between Consumers and Producers.
- Policy III: focus simply on the metrics which are directly optimized by the UTILITY and COMPENSATION adjustments, i.e. NDCG, $\tau_{margin}$. The goal is to understand if optimizing few and adjustments directly correlated with the objectives is an easier task than to optimize for a larger set of objectives, with diverse set of goals.
- Policy IV: focus mostly on customer relevance, with no direct metric aiming for COMPENSATION adjustment: i.e. NDCG, $\tau_{price}$, $\tau_{rating}$. Here we attempt to observe how does the process behave when we disregard both System and Producers objectives.
- Policy V: aims to optimize only partner's fairness objectives, i.e. $GF_{market}$, $GF_{type}$, $GF_{recency}$, $GF_{prod}$, $GF_{brand}$. Here, we disregard any objective regarding Customer and System, thus trying to understand how much can fairness be improved using the proposed approach and what are its effects on all stakeholders.
- Policy VI: tries to find the best overall result for all objectives, thus testing the outcome of using as many objectives as desired: i.e. NDCG, $\tau_{price}$, $\tau_{margin}$, $\tau_{distance}$, $\tau_{rating}$, $\tau_{reviews}$, $GF_{market}$, $GF_{type}$, $GF_{recency}$, $GF_{brand}$, $GF_{prod}$.
- Policy VII: aims to find a good balance between conversion and fairness, while making sure the results are price sensitive, i.e. NDCG, $\tau_{price}$, $GF_{type}$, $GF_{recency}$, $GF_{brand}$. The goal is to understand how does the framework behave without any direct optimization for System objectives.

### 4.3 Meta-labels

Following the procedure from Section 3.3, we have set the following ranges: $w_u \in [0.8, 1.5]$ and $w_c \in [0.3, 1.5]$. The decision has been made purely from domain knowledge and our wish to not deviate considerably from the default settings. Furthermore, considering there are 2 adjustments, the sections defined follow the example presented in Figure 2. Table 1 presents the respective weight ranges per section, with increments of 0.3 and 0.4 for $w_u$ and $w_c$, respectively.

| | low/low | low/high | med/med | high/low | high/high |
|---|---|---|---|---|---|
| $w_u$ | [0.8, 1.1] | [0.8, 1.1] | [1.1, 1.4] | [1.4, 1.7] | [1.4, 1.7] |
| $w_c$ | [0.3, 0.7] | [1.1, 1.5] | [0.7, 1.1] | [0.3, 0.7] | [1.1, 1.5] |

Table 1. UTILITY and COMPENSATION weight ranges per section in the objective space.

Each section is explored with a grid size of $0.1 \times 0.1$, leading to 12 candidate points per section. The best solution per policy is presented in Table 2, using the notation $w_u/w_c$.

| Pol. | low/low | low/high | med/med | high/low | high/high |
|---|---|---|---|---|---|
| I | 0.8/0.3 | 0.8/1.5 | 1.0/0.8 | 1.4/0.3 | 1.4/1.5 |
| II | 0.8/0.3 | 0.8/1.5 | 1.0/0.8 | 1.2/0.3 | 1.2/1.5 |
| III | 0.8/0.6 | 0.8/1.5 | 1.0/1.0 | 1.4/0.6 | 1.4/1.5 |
| IV | 0.8/0.3 | 0.8/1.4 | 1.0/0.7 | 1.4/0.3 | 1.4/1.4 |
| V | 0.8/0.3 | 0.8/1.5 | 1.0/0.7 | 1.2/0.3 | 1.2/1.1 |
| VI | 0.8/0.4 | 0.8/1.5 | 1.0/0.8 | 1.4/0.4 | 1.4/1.5 |
| VII | 0.8/0.3 | 0.8/1.1 | 1.0/0.7 | 1.4/0.3 | 1.4/1.3 |

Table 2. Best solution per policy, with all candidate meta-labels the model can choose from at inference time. These are selected from the ranges presented in Table 1.

The results show there are indeed different candidate preferences, depending on the policy, which validate the procedure employed. However, a couple of interesting results must be highlighted: 1) $w_u = 0.8$ is always used for low/low and low/high sections and medium/medium section always assigns $w_u = 1$, but with different $w_c$. This shows a couple of things: the lowest end of the utility range could be even lower, to allow further discrimination amongst policies (we refrain from doing so in order to ensure a minimum customer relevance score) and the default setting $w_u = w_c = 1$ is not always the ideal scenario across policies - it seems to work in UTILITY but not in COMPENSATION.

### 4.4 Meta-model

The meta-models are fitted using lightGBM [25] and tuned using hyperopt [7]. Notice only this algorithm's performance is reported, since we wish to evaluate the framework's overall behaviour and not find the best tuned model. Thus, we have chosen a model which has provided good performance overall in offline testing, although we do not exclude the hypothesis another model could perform better, depending on the policy employed. Furthermore, each meta-model is compared only against majority voting baseline, which always predicts the most frequent label. This is the only baseline we use because we could not find another direct baseline for this problem. In the simulations, we compare against default settings ($w_u = w_c = 1$).

## 5 RESULTS

### 5.1 Meta-model predictive performance

Table 3 presents the performance of Juggler and baseline models in each policy, across multiple classification predictive performance metrics: precision, recall, F1, False Positive Rate (FPR) and Area Under Curve (AUC).

| Policy | Model | Precision | Recall | F1 | FPR | AUC |
|---|---|---|---|---|---|---|
| I | Juggler | **0.40** | **0.41** | **0.39** | **0.16** | **0.72** |
| | Baseline | 0.06 | 0.25 | 0.10 | 0.25 | 0.5 |
| II | Juggler | **0.35** | **0.37** | **0.35** | **0.18** | **0.68** |
| | Baseline | 0.07 | 0.27 | 0.12 | 0.27 | 0.5 |
| III | Juggler | **0.41** | **0.45** | **0.37** | **0.35** | **0.63** |
| | Baseline | 0.17 | 0.42 | 0.25 | 0.42 | 0.5 |
| IV | Juggler | **0.40** | **0.45** | **0.38** | **0.26** | **0.67** |
| | Baseline | 0.11 | 0.33 | 0.17 | 0.33 | 0.5 |
| V | Juggler | **0.40** | **0.44** | **0.40** | **0.22** | **0.73** |
| | Baseline | 0.13 | 0.36 | 0.19 | 0.36 | 0.5 |
| VI | Juggler | **0.40** | **0.43** | **0.39** | **0.17** | **0.72** |
| | Baseline | 0.09 | 0.29 | 0.13 | 0.29 | 0.5 |
| VII | Juggler | **0.38** | **0.41** | **0.37** | **0.21** | **0.68** |
| | Baseline | 0.09 | 0.30 | 0.14 | 0.30 | 0.5 |

Table 3. Meta-model scores across policies.

The results show Juggler consistently outperforms the baseline in all metrics, showing it is the better option to correctly predict the meta-labels assigned. We shall inspect the value of such predictions in Section 5.3.

### 5.2 Meta-feature importance

Table 4 presents the distribution of meta-features in Juggler for each policy. The procedure counts how frequently each feature appears in tree nodes, with a higher value meaning it is more important. The results are grouped by meta-feature class for readability purposes.

| $\mathcal{F}$ | I | II | III | IV | V | VI | VII |
|---|---|---|---|---|---|---|---|
| Search | 81.60 | 74.35 | 75.20 | 79.43 | 71.38 | 79.77 | 70.79 |
| $f_{RH}$ | 18.10 | 25.65 | 21.80 | 18.55 | 28.62 | 20.23 | 29.21 |
| $f_{RC}$ | 0.30 | 0 | 3.00 | 2.02 | 0 | 0 | 0 |

Table 4. Percent share from each meta-feature class.

The results show the search features are the most frequent features ($> 70\%$) in all Juggler models, an expected result given the segmentation ability of multiple categorical variables - in fact, the destination identifier accounts for the majority of such frequency. On the other hand, ranking correlation meta-features $f_{RC}$ are almost negligible. A justification for this behaviour may come from the fixed number of properties used to create the rankings (i.e. only top 30 properties). Future work should address this issue by considering different thresholds. The ranking histogram meta-features $f_{RH}$ account for $18 - 30\%$ of feature frequency, thus establishing themselves as a valuable meta-feature.

### 5.3 Simulations

Figure 4 presents the distribution of predicted meta-labels, represented by respective sections for readability purposes.

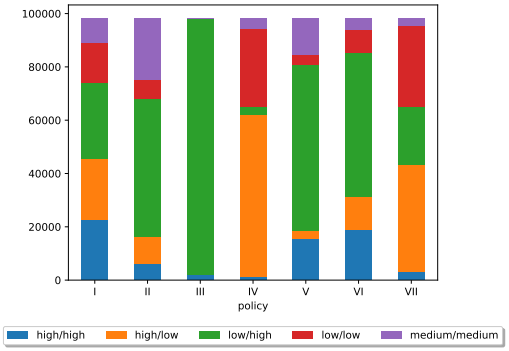


Fig. 4. Predicted section distribution per policy.

We observe most policies take advantage of all 5 sections, which is an indicator this is a suitable solution to define meta-labels. Good examples can be observed in policies I, II, V, VI and VII. Notice also how the medium/medium section is typically one of the sections with less predictions. This is in itself a justification to use dynamic weight prediction, the key contribution of our work.

There are however, some policies with significant class imbalance, most notably policy III: the vast majority of predictions fall under the low/high section. Such problem could potentially be addressed through increasing the granularity of grid-search when defining sections, although we cannot exclude the hypothesis that the policy is ill-defined.

### 5.4 Multi-Stakeholder impact

We inspect now the impact of predictions in terms of the dimensions explored by the policies. Table 5 presents the percent changes against the default method ($w_u = w_c = 1$) in top 30 properties with regards to multiple dimensions.

| Metric | I | II | III | IV | V | VI | VII |
|---|---|---|---|---|---|---|---|
| NDCG | 0.7 | -1.8 | -5.6 | 6.0 | -3.2 | -2.6 | 4.3 |
| Price | 18.7 | 2.7 | -16.2 | -27.0 | -4 | -0.1 | 22.3 |
| Margin | 8.7 | 18.6 | 0.5 | 19.4 | 11.7 | 8.7 | 20.7 |
| Rating | 3.7 | 2.0 | -3.4 | 6.2 | 0.1 | 0.5 | 5.6 |
| Distance | 21.3 | 7.6 | -17 | 27.7 | -1.2 | 1.7 | 22 |
| Reviews | 92.5 | -6.9 | -27.8 | 71.7 | -15.1 | -9.4 | 24.2 |
| Market | -32.8 | -16.6 | 24.3 | -51.3 | -1.7 | -5.4 | -46.2 |
| Production | -14.9 | -7.3 | 11.3 | -23.6 | -1.0 | -2.1 | -20.6 |
| Type | -31.8 | -15.6 | 23.5 | -50 | -1.2 | -4.8 | -44.5 |
| Recency | -28.7 | -14 | 21.3 | -45.1 | -1.2 | -4.2 | -40 |
| Brand | -27 | -13.7 | 20.2 | -42.4 | -1.4 | -4.3 | -38.1 |

Table 5. Percent changes over default ($w_u = w_c = 1$), with positive values meaning an improvement. Underlined values refer to metrics optimized in policy, for improved readability.

The results, show the following:

- Policy I - improves in all metrics in policy, most notably in distance and price. However, the fairness metrics are negatively affected, particularly in property type and sub-market. Although this policy works as expected, the impact is too negative on the Providers side, meaning it will negatively affect our relationship with them and potentially lead to withdrawal from the System.
- Policy II - does not work as expected: it shows a decrease in all objective metrics, specially in terms of fairness. It shows, however, improvement in price, margin, rating and distance. One possible justification is that the current set of adjustments has been designed and tuned from System's point of view, and leads to sub-optimal performance when we try to tune it for a different perspective.
- Policy III - improves in margin, but with negative effect on NDCG. Indirectly, it improves all fairness metrics, but conversion is severely impaired. This behaviour points out to the overall strength of the compensation adjustment, which seems to overpower at times the utility score - to address this issue, one could retry the experiment with more constraints on the compensation adjustment weight range.
- Policy IV - improves in all metrics, plus in margin and reviews. It seems such policy optimizes also for System's objectives, which is an interesting finding. However, it introduces however the most negative effect in fairness metrics across policies - this behaviour may lead, like in Policy I, to risk of Producers leaving the System, which will contribute to negative long term value.
- Policy V - fails in all metrics in policy (and most of the remaining), except for margin. However, the resulting policy is the one which deviates the least from the default ranking - this indicates there may be issues in optimizing for fairness objectives without any specific fairness adjustment included in the equation. It may also mean the existing fairness metrics are not ideal to be optimized in this setup, although such observation requires more experimental data to be supported.

- Policy VI - improves margin, rating and distance but it is worse in all other metrics. The results seem to indicate, like in Policy III, that the subset of weights selected for the compensation weight is not ideal as it allows the compensation adjustment to overpower the remaining adjustments.
- Policy VII - improves NDCG and price, but it degrades substantially in fairness metrics. It also improves ratings, margin, distance and reviews. This policy performs well overall, although it also highlights the issue on the exclusion of fairness based adjustments from the equation.

Overall, the simulation results show a decent behaviour against the policy expectations, with only 2 policies failing on most or all accounts: policies II and V. Such policies have in common the fact that they optimize for all fairness metrics, which may be an indication that the proposed Group Fairness function in Equation 6 is not the most suitable definition for this scope. Also, it could mean that the non-existence of a fairness adjustment to weight using the meta-model, would make such optimization troublesome. It would be interesting to try different fairness definitions in future work to understand whether this issue can be addressed in such policies.

Another interesting observation comes from the fact that not all policies are easily optimizable. For instance, while policies I and IV optimize all metrics, policies III and VI are able to optimize only in a few objectives. This happens because: 1) the hidden adjustments in Equation 10 have great impact on the overall results (these are most noticeable when the weights predicted are lower than the default value) and 2) the predicted weights affect all items in a search equally, meaning it has limited impact on how much can be optimized. Both points can be addressed by including the remaining adjustments under this framework.

Regarding stakeholder's impact, it seems policies are better at improving recommendations for Customers and System than for Providers: in fact, only in policy III are fairness metrics greatly improved, even though they are not included in the policy. This means the current recommendation procedure can only increase fairness by increasing margin. To address this issue, future work could weight directly fairness adjustments through Juggler using global weights or weight each property individually across all adjustments - this would allow more granular corrections using Juggler and a potential better approximation to all stakeholder's objectives.

## 6 CONCLUSIONS

This paper has presented a Meta-Learning approach to address the Multi-Stakeholder recommendation problem. We take advantage of historical transactions and simulations to understand which are the best weights per search, given a specific policy. The results show the models have high predictive performance and simulations have shown it is able to successfully improve multiple objectives, depending on the chosen policy. There are however many issues to be improved, namely: replace manually defined meta-features with embeddings, to identify more and better metrics to be optimized in policies, to use Reinforcement Learning models able to perform exploration of objective vectors per search (for instance, Multi-Armed bandits) and to improve the policy definition mechanism using constraints and/or the definition of primary and secondary objectives, in order to attempt to clarify assignment of best labels.

## REFERENCES


[1] Himan Abdollahpouri, Gediminas Adomavicius, Robin Burke, Ido Guy, Dietmar Jannach, Toshihiro Kamishima, Jan Krasnodebski, and Luiz Pizzato. 2020. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction* 30, 1 (may 2020), 127–158.

[2] Himan Abdollahpouri and Robin Burke. 2019. *Multi-stakeholder Recommendation and its Connection to Multi-sided Fairness.* Technical Report. arXiv:1907.13158v1

[3] Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. 2017. Recommender systems as multistakeholder environments. In *UMAP 2017 - Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 347–348.

[4] Himan Abdollahpouri and Steve Essinger. 2017. Multiple stakeholders in music recommender systems. arXiv:1708.00120

[5] Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems* 3, 1 (2012), 1–17.

[6] Amos Azaria, Avinatan Hassidim, Sarit Kraus, Adi Eshkol, Ofer Weintraub, and Irit Netanely. 2013. *Movie Recommender System for Profit Maximization.* Technical Report.

[7] J. Bergstra, D. Yamins, and D. D. Cox. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA). JMLR.org, I–115–I–123.

[8] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta. 2009. *Metalearning: Applications to Data Mining* (1 ed.). Springer.

[9] Robin Burke, Nasim Sonboli, and Aldo Ordoñez-Gauger. 2018. *Balanced Neighborhoods for Multi-sided Fairness in Recommendation.* Technical Report.

[10] Andrew Collins, Joeran Beel, and Dominika Tkaczyk. 2018. One-at-a-time: A Meta-Learning Recommender-System for Recommendation. arXiv:1805.12118

[11] Andrew Collins, Laura Tierney, and Joeran Beel. 2020. *Per-Instance Algorithm Selection for Recommender Systems via Instance Clustering.* Technical Report.

[12] Tiago Cunha, Carlos Soares, and André C.P.L.F. Carvalho. 2017. Metalearning for Context-aware Filtering: Selection of Tensor Factorization Algorithms. In *Proceedings of the 11th ACM Conference on Recommender Systems*. ACM, 14–22.

[13] T. Cunha, C. Soares, and A.C.P.L.F. de Carvalho. 2018. Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. *Information Sciences* 423 (2018).

[14] Tiago Cunha, Carlos Soares, and André C.P.L.F. de Carvalho. 2016. Selecting Collaborative Filtering algorithms using Metalearning. In *European Conference on Machine Learning and Knowledge Discovery in Databases*. 393–409.

[15] Tiago Cunha, Carlos Soares, and André C P L F de Carvalho. 2018. CF4CF: Recommending Collaborative Filtering Algorithms Using Collaborative Filtering. In *Proceedings of the 12th ACM Conference on Recommender Systems*. 357–361.

[16] Michael Ekstrand and John Riedl. 2012. When Recommenders Fail: Predicting Recommender Failure for Algorithm Selection and Combination. In *Proceedings of the 6th ACM Conference on Recommender Systems*. ACM, 233–236.

[17] Michael D. Ekstrand, Robin Burke, and Fernando Diaz. 2019. Fairness and discrimination in recommendation and retrieval. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM, Inc, 576–577.

[18] Michael T M Emmerich and André H Deutz. 2018. A tutorial on multiobjective optimization: fundamentals and evolutionary methods. *Natural Computing* 17 (2018), 585–609.

[19] Ruoyuan Gao and Chirag Shah. 2019. How fair can we go: Detecting the boundaries of fairness optimization in information retrieval. In *ICTIR*. ACM, 229–236.

[20] Jonathan L Herlocker, Joseph a. Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22, 1 (2004), 5–53.

[21] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. 2008. Evolutionary Many-Objective Optimization: A Short Review. In *Proc. of 2008 IEEE Congress on Evolutionary Computation*. 2424–2431.

[22] Dietmar Jannach and Gediminas Adomavicius. 2016. Recommendations with a purpose. In *Proceedings of the 10th ACM Conference on Recommender Systems*. 7–10.

[23] A Kalousis. 2002. *Algorithm Selection via Meta-Learning.* Ph.D. University of Geneve, Department of Computer Science.

[24] Jorge Kanda, Andre de Carvalho, Eduardo Hruschka, Carlos Soares, and Pavel Brazdil. 2016. Meta-learning to select the best meta-heuristic for the Traveling Salesman Problem: A comparison of meta-features. *Neurocomputing* 205 (2016), 393–406.

[25] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc.

[26] Bingdong Li, Jinlong Li, Ke Tang, and Xin Yao. 2015. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv* 48, 13 (2015).

[27] Wei Lu, Shanshan Chen, Keqian Li, and Laks V.S. Lakshmanan. 2014. Show me the money: Dynamic recommendations for revenue maximization. In *Proceedings of the VLDB Endowment*. 1785–1796.

[28] Rishabh Mehrotra and Benjamin Carterette. 2019. Recommendations in a marketplace. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 580–581.

[29] Rishabh Mehrotra, James McInerney, Hugues Bouchard, Mounia Lalmas, and Fernando Diaz. 2018. Towards a Fair Marketplace: Counterfactual Evaluation of the Trade-off between Relevance, Fairness and Satisfaction in Recommendation Systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 2243–2251.

[30] Mustafa Mısır and Michèle Sebag. 2017. Alors: An algorithm recommender system. *Artificial Intelligence* 244, 244 (2017), 291–314.

[31] Phong Nguyen, John Dines, and Jan Krasnodebski. 2017. A multi-objective learning to re-rank approach to optimize online marketplaces for multiple stakeholders. arXiv:1708.00651

[32] Phong Nguyen, Jun Wang, Melanie Hilario, and Alexandros Kalousis. 2012. Learning Heterogeneous Similarity Measures for Hybrid-Recommendations in Meta-Mining. *IEEE International Conference on Data Mining* (2012), 1026–1031.

[33] T Pereira, T Cunha, and C Soares. 2020. $\mu$ − cf2vec: Representation Learning for Personalized Algorithm Selection in Recommender Systems. In *2020 International Conference on Data Mining Workshops*. IEEE Computer Society, 181–188.

[34] John Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 65–118.

[35] Mario Rodriguez, Christian Posse, and Ethan Zhang. 2012. Multiple objective optimization in recommender systems. In *Proceedings of the 6th ACM Conference on Recommender Systems*. 11–18.

[36] André Luis Debiaso Rossi, André Carlos Ponce De Leon Ferreira de Carvalho, Carlos Soares, and Bruno Feres de Souza. 2014. MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing* 127 (March 2014), 52–64.

[37] Kate Smith-Miles. 2008. Cross-disciplinary perspectives on meta-learning for algorithm selection. *Comput. Surveys* 41, 1 (Dec. 2008), 1–25.

[38] Carlos Soares, Pavel B Brazdil, and Petr Kuba. 2004. A Meta-Learning Method to Select the Kernel Width in Support Vector Regression. *Machine Learning* 54, 3 (2004), 195–209.

[39] Joaquin Vanschoren. 2010. *Understanding machine learning performance with experiment databases*. Ph.D. Dissertation. Katholieke Universiteit Leuven.

[40] Joaquin Vanschoren. 2018. Meta-Learning: A Survey. *CoRR* abs/1810.03548 (2018). arXiv:1810.03548

[41] Christian Von Lücken, Benjamín Barán, and Carlos Brizuela. 2014. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications* 58 (2014), 707–756.

[42] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. In *Proceedings of the 26th Annual Conference on Learning Theory (Proceedings of Machine Learning Research, Vol. 30)*. PMLR, Princeton, NJ, USA, 25–54.

[43] David Wolpert and William Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (April 1997), 67–82.

[44] Yong Zheng. 2019. Multi-Stakeholder Recommendations: Case Studies, Methods and Challenges. In *Proceedings of the 13th ACM Conference on Recommender Systems*. ACM.

[45] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. 2011. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation* 1 (2011), 32–49.