Faceted Search with Object Ranking and Answer Size Constraints

Konstantinos Manioudakis

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science and Engineering

University of Crete
School of Sciences and Engineering
Computer Science Department
Voutes University Campus, 700 13 Heraklion, Crete, Greece

Thesis Advisor: Associate Prof. *Yannis Tzitzikas*

This work has been performed at the University of Crete, School of Sciences and Engineering, Computer Science Department.

The work has been supported by the Foundation for Research and Technology - Hellas (FORTH), Institute of Computer Science (ICS).

University of Crete
Computer Science Department

Faceted Search with Object Ranking and Answer Size Constraints

Thesis submitted by
Konstantinos Manioudakis
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Konstantinos Manioudakis

Committee approvals: _____
Yannis Tzitzikas
Assistant Professor, University of Crete
Thesis Supervisor

_____
Dimitris Plexousakis
Professor, University of Crete
Committee Member

_____
Kostas Magoutis
Associate Professor, University of Crete
Committee Member

Departmental approval: _____
Antonis Argyros
Professor, University of Crete
Director of Graduate Studies

Heraklion, February 2020

Faceted Search with Object Ranking and Answer Size Constraints

Abstract

Faceted Search is a widely used interaction scheme in e-commerce, digital libraries, and recently also in Linked Data. Surprisingly, object ranking in the context of Faceted Search is not well studied in the literature. In this thesis we propose an extension of the model with two parameters that enable specifying the desired answer size and the granularity of the sought object ranking. These parameters allow tackling the problem of too big or too small answers and can specify how refined the sought ranking should be. Then we provide an algorithm that takes as input these parameters and by considering the hard-constraints (filters) and the soft-constraints (preferences) that the user has formulated while interacting with the system, as well as the statistical properties of the underlying dataset (through various frequency-based ranking schemes), it produces an object ranking that tries to satisfy these parameters. Then we present extensive simulation-based evaluation results (over several datasets) which provide evidence that the proposed model also improves the answers and reduces the user's average cost (9.7% and 7.4%) as well as the maximum cost (21.8% and 26.8%), when applying hard and soft constraints respectively. Finally, we propose the required GUI extensions and present an implementation of the model.

Πολυεδρική Αναζήτηση με Κατάταξη Αντικειμένων και Περιορισμούς Μεγέθους Απάντησης

Περίληψη

Η Πολυεδρική Αναζήτηση (Faceted Search) είναι ένα ευρέως χρησιμοποιούμενο υπόδειγμα διαλογικής αναζήτησης στο ηλεκτρονικό εμπόριο, τις ψηφιακές βιβλιοθήκες, και πρόσφατα στα Διασυνδεδεμένα Δεδομένα. Απροσδόκητα, η κατάταξη αντικειμένων στο πλαίσιο της Πολυεδρικής Αναζήτησης δεν είναι καλά μελετημένη στη βιβλιογραφία. Σε αυτήν την εργασία προτείνουμε μία επέκταση του μοντέλου αλληλεπίδρασης με δύο παραμέτρους που καθιστούν εφικτό τον καθορισμό του μεγέθους της απάντησης και της λεπτομερειακότητας της επιθυμητής κατάταξης αντικειμένων. Αυτές οι παράμετροι επιτρέπουν την αντιμετώπιση του προβλήματος των υπερβολικά μεγάλων ή υπερβολικά μικρών απαντήσεων και μπορούν να καθορίζουν πόσο εκλεπτυσμένη πρέπει να είναι η ζητούμενη κατάταξη. Έπειτα παρέχουμε έναν αλγόριθμο που δέχεται ως είσοδο αυτές τις παραμέτρους και λαμβάνοντας υπ' όψιν τους περιορισμούς που έχει εκφράσει διαλογικά ο χρήστης, είτε είναι αυστηροί-περιορισμοί (φίλτρα), ή χαλαροί-περιορισμοί (προτιμήσεις), καθώς και τις στατιστικές ιδιότητες του υποκείμενου συνόλου δεδομένων (μέσω διάφορων μεθόδων κατάταξης βασισμένων στη συχνότητα), παράγει μία κατάταξη αντικειμένων που προσπαθεί να ικανοποιήσει αυτές τις παραμέτρους. Έπειτα παρουσιάζουμε εκτεταμένα αποτελέσματα από αξιολόγηση μέσω προσομοίωσης σε πολλά σύνολα δεδομένων, τα οποία καταδεικνύουν ότι το προτεινόμενο μοντέλο βελτιώνει τις απαντήσεις και μειώνει το μέσο κόστος του χρήστη (9.7% και 7.4%) καθώς και το μέγιστο κόστος του χρήστη (21.8% και 26.8%), όταν εκφράζονται αυστηροί και χαλαροί περιορισμοί αντίστοιχα. Τέλος, προτείνουμε τις αναγκαίες επεκτάσεις της γραφικής διεπαφής χρήστη και παρουσιάζουμε μια υλοποίηση του μοντέλου.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω θερμά τον επόπτη καθηγητή μου κ. Γιάννη Τζίτζικα για την ορθή καθοδήγηση και ουσιαστική συμβολή του στην ολοκλήρωση της παρούσας μεταπτυχιακής εργασίας. Ακόμη θέλω να εκφράσω τις ευχαριστίες μου στους κ. Δημήτρη Πλεξουσάκη και κ. Κώστα Μαγκούτη για την προθυμία τους να συμμετέχουν στην τριμελή επιτροπή. Δεν θα μπορούσα να παραλείψω τις ευχαριστίες μου στους συναδέλφους του εργαστηρίου Παναγιώτη Παπαδάκο, Γιάννη Μαρκετάκη και Μιχάλη Μουνταντωνάκη, που ήταν πάντα πρόθυμοι να με βοηθήσουν. Επίσης θα ήθελα να ευχαριστήσω το Ινστιτούτο Πληροφορικής του Ιδρύματος Τεχνολογίας και Έρευνας για την πολύτιμη υποστήριξη σε υλικοτεχνική υποδομή και τεχνογνωσία, καθώς και για την υποτροφία που μου προσέφερε καθ΄ όλη τη διάρκεια της μεταπτυχιακής μου εργασίας. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για τη συμπαράσταση και την υποστήριξη που μου έδωσαν όλα αυτά τα χρόνια.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Faceted Search (FS) is the established information access scheme in e-commerce for more than 10 years [26, 31]. It is widely used in areas such as digital libraries [29, 15], semantic web [10], Linked Data [20] and Knowledge Graphs in general [9]. FS is essentially a *session-based* interactive method for *gradual query formulation* (commonly over a multidimensional information space) through *simple clicks* that offers to the user an *overview* of the result set (groups and count information) and *never leads to empty results sets*. At each state of the interaction, users explore the set of objects that satisfy the various restrictions they have specified up to that point. The result set is called the *focus*. Objects inside the focus are *unranked*, e.g. when the user examines an online prospectus for buying a new camera, or *ranked*, e.g. when the user explores available hotels which are ordered with respect to price, star rating or other criteria (default or user specified in the form of preferences as in the case of Preference-enriched Faceted Search - PFS [35]). The focus is ranked also in cases where FS is applied after a keyword search query, e.g. as in Google Scholar. Although object ranking in FS is already used in commercial systems (as mentioned earlier), the scientific literature on this topic is not extensive. Previous research has mainly focused on *facet ranking*, i.e. on methods for deciding which, and in what order, facets to present, an issue that is important if the domain of the dataset is wide (e.g. faceted search over DBpedia). In this thesis we elaborate on the *objects ranking*, in the context of Faceted Search in domain specific settings, where the set of useful facets is known (as in product buying, bookings, etc). Even if Faceted Search solves the problem of empty answers, it cannot solve the problem of *too big* or *too small* answers. In such cases, the user has to make extra actions (for restricting a big answer or for relaxing a too small answer).

To tackle these problems, in this thesis we propose an extension of the FS model that is enriched with two parameters that enable specifying the desired answer size and the granularity of the sought object ranking. These parameters allow tackling the problem of *too big* or *too small* answers and can specify *how refined* the sought ranking should be. Then we provide an algorithm that takes as input these parameters and by considering the hard-constraints (filters), the

soft-constraints (preferences), as well as the statistical properties of the dataset, produces an object ranking that satisfies these parameters.

Table 1.1: An example dataset of hotels

Object	Location	Stars	Price
o_1	Hyogo	4	308
o_2	Hyogo	4	226
o_3	Hyogo	4	265
o_4	Kyoto	4	218
o_5	Hyogo	4	402
o_6	Hyogo	3	209
o_7	Kyoto	4	293
o_8	Hyogo	4	460
o_9	Hyogo	4	208
o_{10}	Hyogo	3	396
o_{11}	Hyogo	5	528
o_{12}	Kyoto	4	81

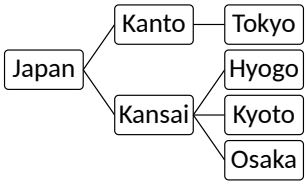

Figure 1.1: Hierarchy for Location facet

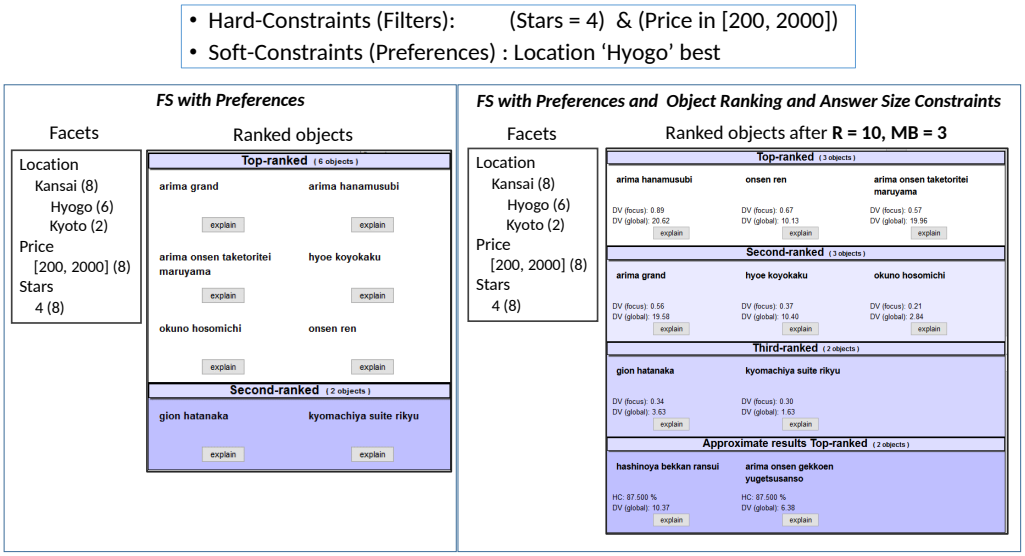

Figure 1.2: Left: A typical FS response with filters and preferences. Right: The response of the extended FS with *approximate results* and *more refined ranking*.

To grasp the idea, Table 1.1 shows a small set of hotels $Obj = \{o_1, \ldots, o_{12}\}$ described by three facets $F = \{Location, Stars, Price\}$ where the terms of the facet Location are hierarchically organized as shown in Figure 1.1. The left side of Figure 1.2 sketches the GUI of a typical FS system that shows the 8 hotels that satisfy the hard-constraints (filters) that the user has selected (in our case 4 stars and price in [200, 2000]), and notice that these hotels are partitioned in two buckets based on the soft-constraints (preferences) that the user has specified (in our case the user has expressed a preference on the location Hyogo). Now the right side of Figure 1.2 sketches the GUI according to the extended model that

we introduce, where we assume that the user has asked for 10 objects, and for a more refined ranking, e.g. that no bucket should contain more than 3 hotels. We can see that two more hotels have appeared (approximate results) and that none of the blocks of the new focus contains more than 3 hotels.

To the best of our knowledge, no other work has focused on the problem of *too big* or *too small* answers and on the *granularity of object ranking* in the context of Faceted Search. To tackle these issues several questions arise including: how to consider in the object ranking all kinds of input (filters, preferences as well as the statistical properties of the datasets), how to evaluate (in a cost effective and repeatable way) whether such extensions improve the interaction with the user, and how to enrich the interaction with rank explanation services. Issues of system efficiency and optimization are out of scope of this thesis (however we do report efficiency results).

In a nutshell, the main contributions of this work are: (a) the extension of FS with two parameters expressing the desired ranking and size properties of the answer, (b) the formulation of the corresponding object ranking problem in a context that assumes both hard- and soft-constraints, and the discussion of the solvability of the problem, (c) the algorithm `SmartFSRank` for producing the object ranking that is based on the factorization of the problem into two simpler sub-tasks, as well as on the adoption of frequency-based ranking schemes, (d) the description of a simulation-based evaluation framework for evaluating the impact of such object ranking methods, and (e) extensive simulation-based evaluation results. The main finding is that the extended model apart from being customizable to answer size and ranking granularity constraints (that enables tackling the problem of too small or big answers), it improves the answers and reduces the average navigation cost, as evidenced by the simulation based evaluation over five datasets from different domains.

This thesis is organized as follows: Chapter 2 presents the background and an overview of the related work. Chapter 3 describes the data representation and the baseline interaction, introduces the extended model and provides the algorithms for realizing that model. Chapter 4 presents simulation-based procedures and results, while Chapter 5 discusses the implementation of the model and its efficiency, the extensions of the GUI, as well as a comparison with related systems. Finally, Chapter 6 concludes this thesis and identifies issues for future work and research.

Parts of this work have been published in the following papers:

1. (published) Extending Faceted Search with Automated Object Ranking. In Metadata and Semantic Research, Emmanouel Garoufallou, Francesca Fallucchi, and Ernesto William De Luca (Eds.). Springer International Publishing, Cham, 223–235, 2019.

2. (on submission) Faceted Search with Object Ranking and Answer Size Constraints. In Transactions on Information Systems, ACM, 2020

Chapter 2

Context and Related Work

Section 2.1 discusses the background and Section 2.2 the related work.

2.1 Background

2.1.1 Faceted Search

Faceted Search (or *Faceted Exploration*), is a widely used interaction scheme for Exploratory Search. It is the de facto query paradigm in e-commerce [26, 31] and in digital libraries [29, 15]. It is also used for exploring RDF Data (e.g. see [34] for a recent survey, and [20] for a recent system), as well as general purpose knowledge graphs [9]. Faceted exploration can facilitate web search, e.g. in the news domain [1]. It has also been applied for automatically summarizing web search results, e.g. [14]. Informally we could define it as a *session-based interactive method for query formulation (commonly over a multidimensional information space) through simple clicks that offers an overview of the result set (groups and count information), never leading to empty results sets.*

2.1.2 Preference-enriched Faceted Search

Although *preferences* have been studied in the context of databases, e.g. see [28] for a survey, and lately in query languages for RDF e.g. [24, 30, 25, 27]), in Faceted Search systems there are only a few related works. *Preference-enriched Faceted Search* (for short PFS), is an extension of FS that supports *preferences*, i.e. apart from actions for specifying filters, it offers actions that allow the user to rank facets, values, and objects using *best, worst, preferTo* actions (i.e. relative preferences), *aroundTo* actions (over a specific value), and other criteria (see [35]). Apart from "primitive" preferences, the user is able to *compose* object related preference actions, using *Priority, Pareto, Pareto Optimal* (i.e. skyline) and other. The distinctive features of PFS is that it allows expressing preferences over attributes whose values can be hierarchically organized (and/or multi-valued), it

supports preference inheritance, and it offers scope-based rules for resolving automatically the conflicts that may arise. As a result, users are able to restrict their current focus by using the faceted interaction scheme (hard restrictions) that lead to non-empty results, and rank according to preference the objects in the focus. Recently, PFS has been used in various domains, e.g. for offering a flexible process for the identification of fish species [32], as a Voting Advice Application [33] and it has been expanded with geographic anchors for being appropriate for the exploration of datasets that contain also geographic information [18]. PFS has also been used in the context of spoken dialogue systems [23].

2.2 Related Work

There is related work from several areas including FS systems, Databases, and Learning to Rank approaches. An overview of the related problems and approaches is shown in Fig. 2.1.

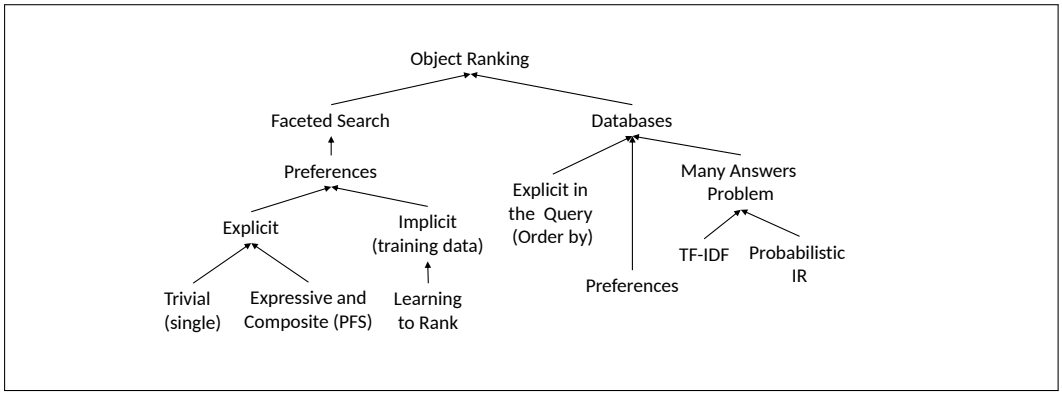

Figure 2.1: An overview of problems and approaches in related work

2.2.1 In Faceted Search Systems

The concept of automatic ranking in faceted search is not recent, and there are numerous approaches, as regards *what* to rank, and *how* to rank, as discussed in the survey [34]. Past research has primarily focused on methods only for *facet ranking*, i.e. for deciding in what order to place the facets. In many works (e.g [12, 11]), the proposed methods depend on the frequency of facet values. The technique described in [37] dynamically ranks the facets depending on the query. In addition, they define different metrics for numeric and qualitative facets, and also recommend ordering the values of each facet with respect to their frequency in descending order. Now [7], relies on a different technique based on set-cover, to produce a ranking of facets. In other works, various metrics have been utilized such as facet *navigational cost* [17] (i.e. how many facet values are available), and

facet *balance* [21] (i.e how evenly the facet values are distributed). Lately, methods for selecting facets from knowledge graphs have also been investigated [9].

Object ranking (in FS systems) is already used in commercial systems. For instance, the online hotel platform `booking.com` offers a ranking method based on a number of properties. However, this topic has not been studied in the scientific literature extensively. In [26] (Chapter 9) a ranking method based on facet values is briefly described through an e-shopping example. According to PFS, and the system Hippalus [22], users may define the objects' ranking by formulating preference actions. In the context of PFS, [33] introduced a method for quantifying the degree of match between an object and the user's preference actions. This aims at showing the user how positive the top-ranked objects are. In the same context, [23] introduces a number of features, specifically, selectivity and entropy, that are exploited for ranking the applicable facets at each point of the user session.

2.2.2 In Databases

Automatic ranking is an issue that has also been studied in the field of databases. A related problem, called the Many Answers Problem, appears when the result tuples of a query are too many and no ordering has been specified (no ORDER BY clause). One of the first approaches for this issue [3], relies on the well established IR framework of TF-IDF weights and cosine similarity.

An alternative approach for the Many Answers Problem [5], is based on Probabilistic Information Retrieval. Specifically, they consider the attributes not specified in the query and measure two scores: a *global score* which reflects the global importance of unspecified attributes and a *conditional score* which captures the strengths of dependencies between specified and unspecified attributes. The estimation of these scores is done automatically from a workload of *past queries*. An evaluation with users revealed higher quality in comparison to the approach in [3].

Now [4] proposed a solution that aims in requesting only a few constraints from the user in order to drill down to a single tuple from a set of ranked or unranked query results. They base their approach on the *decision tree model*. Specifically, inner nodes represent attributes, leaf nodes are tuples and edges are labelled by attribute values. During the process, attributes are chosen dynamically, with regard to their *selectivity* and *information gain*. The target is to choose the minimum number of attributes during this process, until the desired tuple is reached. Finally, another line of works that deal with tuple ranking in databases, are those that support explicit preferences, see [28] for a survey.

2.2.3 Learning to Rank Approaches.

Lately, in the field of Information Retrieval, there is an increasing interest on adopting Machine Learning methods for document ranking [19]. These approaches are based on creating a ranking model, by training a ML model on user data, typically collected from past search queries (e.g. number of document clicks, user

ratings, etc.) A Learning to Rank approach for faceted search is proposed in [36]. That work aims in optimizing the weights of a facet based TF-IDF scoring formula. Specifically, documents and queries are expressed as sets of facet-value pairs. After training the learning methods on user data, they are able to estimate the optimal weights of the formula, given a user query and a list of previously judged documents.

Chapter 3

The Proposed Approach

Section 3.1 describes the data representation and the baseline interaction, Section 3.2 introduces the extended model and Section 3.3 provides the algorithms for realizing that model.

3.1 Modeling the Data Space and the Interaction of Faceted Search (and Preference-enriched Faceted Search)

Before introducing the extended model we first have to model the context, i.e. the structure of the underlying data space (in §3.1.1), as well as, the basics of the interaction of FS and PFS (in §3.1.2), also for reasons of self-containedness.

3.1.1 The Data Space

We consider datasets in the form multidimensional data with hierarchically organized values and multi-valued attributes.

Definition 1 (Data Space)**.** *Let $Obj = \{o_1, \ldots, o_n\}$ be the set objects. We have K facets $F = \{F_1, \ldots, F_K\}$ each associated with a taxonomy $(T_i, \leq_i)$ where T_i is a set of terms, or values, while $\leq_i$ is a (possible empty) partial order over T_i enabling to organize the values of T_i hierarchically. Each object $o \in Obj$ is described by associating it with one or more values from each facet. Let $\vec{o_i}$ denote the description of o_i in that space, i.e. $\vec{o_i} = (o_{i1}, \ldots, o_{iK})$ where $o_{ij} \in T_j \cup \{\epsilon\}$, where ϵ denotes the missing value, and let $\vec{Obj}$ be the set of descriptions of all objects in Obj.* $\diamond$

Note that the above definition of taxonomy, includes linearly ordered terms, i.e. it captures numerically-valued facets. Consequently, this definition captures datasets like the one in Table 1.1.

3.1.2 Modeling the Interaction (sessions of hard and soft constraints)

In the following we assume the data space defined previously. In the interaction scheme of Faceted Search the user explores the data space and expresses gradually a set of hard constraints.

Definition 2 (Hard Constraint)**.** *A* hard constraint *hc is any conjunction of terms over* $T = T_1 \cup \ldots \cup T_K$. $\diamond$

The user through a GUI formulates such conjunction with simple clicks. For reasons of space, we do not describe here the GUI, nor the user actions of Faceted Search, nor how the clicks define the *hc* (the interested reader can refer to [34]). In general, any Boolean expression over T is a hard constraint, however conjunctions are most widely used.

Definition 3 (Extension)**.** *The* extension of a hard constraint *hc, denoted by* $E(hc)$*, is the subset of Obj that satisfies the conjunction hc. The extension of a term* $t_i \in T_i$ *is defined in a way for considering the semantics of taxonomies (if they exist), i.e. it equals the objects having that term or a narrower term, formally:* $E(t_i) = E_{tmp}(t_i) \cup \{E_{tmp}(t_x) \mid t_x \leq t_i\}$ *where* $E_{tmp}(t_i) = \{o_j \in Obj \mid o_{ji} = t_i\}$*. The extension of a boolean expression is defined straightforwardly (by interpreting conjunction with* $\cap$*, disjunction with* $\cup$*, and negation with set-minus).* $\diamond$

In our running example, the extension of $(Stars = 4) \wedge (200 \leq Price \leq 2000)$ is $\{o_1, o_2, o_3, o_4, o_5, o_7, o_8, o_9\}$.

Definition 4 (Soft Constraint)**.** *A* soft constraint *sc is any set of preference actions of the preference language defined in [35].* $\diamond$

In brief these preference actions define a preference relation (a binary relation) over each T_i, denoted by $\succ_i$.

In our running example, the preference action $Hyogo \succ_{Location} Kyoto$, results in the following preference bucket order: $\langle (Hyogo),\ (Kyoto),\ (\{Tokyo, Osaka\})\rangle$.

It is also worth noting that, preference inheritance is supported. For example, if the user had issued the preference action $Kansai \succ_{Location} Kanto$, then the resulting bucket order would be $\langle (\{Hyogo, Kyoto, Osaka\}),\ (Tokyo)\rangle$.

There are preference actions for *composing* these in order to define a preference relation over all possible elements of the data space i.e. over $V = T_1 \times \ldots \times T_K$. Note that since an object can be associated with more than one value from a facet (not in this example), it is more precise to define V as the Cartesian product $\mathcal{P}(T_1) \times \ldots \times \mathcal{P}(T_K)$ where $\mathcal{P}(T_i)$ denotes the powerset of T_i.

Now since the description of the objects $\vec{Obj}$ is a subset of V (i.e. $\vec{Obj} \subseteq V$), a set of soft constraints sc defines a preference relation over Obj denoted as $(Obj, \succ_{sc})$. From $(Obj, \succ_{sc})$ a *bucket order* of Obj, i.e. a linear order of subsets of Obj, is produced though topological sorting (see [35] for details). In our running example,

the preference action $Hyogo \succ_{Location} Kyoto$ results in the following bucket order of the objects in focus: $\langle(\{o_1, o_2, o_3, o_5, o_8, o_9\}),\ (o_4, o_7,)\rangle$.

Let denote the bucket order by BO_{sc}, i.e. $BO_{sc} = \langle b_1, \ldots, b_Z\rangle$ where b_1 contains the most preferred objects, while b_Z the least preferred. All sets b_i $(1 \leq i \leq Z)$ form a *partition* of Obj (i.e. they are pairwise disjoint and their union is Obj). The number of blocks Z ranges between 1 and $|Obj|$. Obviously, if $Z = |Obj|$ then the ranking forms a linear order of Obj, while if $Z = 1$ then all objects are equally ranked (this is true if the sc is empty, e.g. at the beginning of the interaction).
Equivalently, we can say that the aforementioned approach ranks the objects i.e. it computes a function $r(sc, \vec{o}) \in [1..Z]$ where $Z \leq |Obj|$, assigning to each object a natural number (the index of the block to which it belongs, e.g. the objects in the most preferred block receive rank equal to 1).
In PFS, the user explores the data space and expresses gradually a set of hard and soft constraints. At each state of the interaction, the user gets those objects that satisfy the formulated hard constraints hc, ordered in blocks according to the expressed preferences sc. This is the essential part of the model of PFS [35] and it is implemented in the system `Hippalus`[22].

Definition 5 (The Answer given Hard and Soft Constraints)**.** *Given a hard constraint hc and a soft constraint sc,* the answer *according to the PFS interaction, is the set of objects $E(hc)$ ordered by the restriction of $\succ_{sc}$ on $E(hc)$, i.e. $(E(hc), \succ_{sc\ |E(hc)})$.* $\diamond$

Definition 6 (User Session)**.** *A* user session *us is a series of actions, $s = \langle a_1, \ldots, a_n\rangle$ where each a_i is a hard or a soft constraint. Let $hc(us)$ denote the hard constraints in us, and $sc(us)$ denote the soft constraints in us.* $\diamond$

3.2 The Extended Model

Parameters of the Extended Model. This thesis proposes extending the model with two parameters

- MB: Maximum Block size. E.g. if $MB = 1$ then the system should return a linear order of objects, if $MB = 2$ the answer should not contain ties between more than 2 objects.
- R: number of requested objects.

With these two parameters several requirements can be tackled:

- Too many objects: R forces the system to rank the objects for returning the best R objects.
- Too few objects: R forces the system to also return approximate objects
- Arbitrary order: MB forces the system to rank the objects so that no block has more than MB objects, and in this way the rank is not arbitrary

Characterizing a bucket order L. Let L be a bucket order $L = \langle b_1, \ldots, b_z\rangle$, i.e. a possible ranked answer, and let $objects(L)$ denote the set of objects that occur in L. The answer L could be characterized according to various criteria:

- **HCsat**. We could say that L satisfies the hard constraints hc, if $objects(L)$ are exactly those that satisfy hc, i.e. $objects(L) = E(hc)$
- **SCsat**. We could say that L satisfies the soft constraints sc (i.e. it respects the preference order), if $L \supseteq \succ_{sc|E(hc)}$. This means that the order relationships of L contain all order relationships of the preference relation $\succ_{sc}$ that involve objects in the focus (i.e. objects in $E(hc)$). In other words, automatic ranking is used only for ranking the objects in each block of the preference order (it never "violates" the blocks, it just adds relationships, and these relationships do not create any cycle).
- **MBsat.** We could say that L satisfies a *maximum allowable block size* MB, if $|b_i| \leq MB$ for each $1 \leq i \leq z$.
- **Rsat.** We could say that L satisfies R, if L contains exactly R objects.

It is not hard to see that it is not always possible to find an L that satisfies all of the above criteria. For instance if the objects that satisfy the hc are less than R, i.e. $|E(hc)| < R$, then the system should either return less objects (sacrificing Rsat), or should try to return R objects by extending $E(hc)$ with the $R - |E(hc)|$ in number "closest" objects (sacrificing HCsat). On the other extreme, if those that satisfy the hc are more than R, then the system should rank them and return the best of them. This is another case of an L that is not HCsat, since it contains less objects than those satisfying hc. A problem statement that includes more requirements follows:

Definition 7 (Problem Statement)**.** *Given a user session us with hard and soft constraints, a parameter R specifying the number of desired objects, a parameter MB specifying the maximum allowable block size, compute and return to the user the "best" (with respect to HCsat, SCsat, MBsat, Rsat) answer L.*

What remains is to clarify what "best" means. A first objective is to produce one or more L that satisfy all the criteria if possible. In case there are more than one, then a method to select one of them is needed. To this end, dataset statistics and other metrics (as we have seen in the related work) can be used. For example, frequently or rarely occurring values could be promoted. If there is no L that satisfies all hard constraints, then one that "better approximates" a bucket order that satisfies them all needs to be found. This issue is elaborated in the next section.

3.3 The `SmartFSRank` Ranking Method

Here, an algorithm is defined that provides a solution to the problem statement (as defined in Def. 7). In general the algorithm exploits PFS, if supported, and it tries to satisfy R by ranking and approximate matching, and MB through ranking based on statistical properties of the data. Note that the algorithm can be applied

even if PFS is not supported (in that case there is one block, i.e. $z = 1$). In brief, the algorithm SmartFSRank, Alg. 1, first tries to satisfy hc (line 1), and then sc (Part 1, line 2) by exploiting the PFS-based ranking method. Then it tries to satisfy R (Part 2, lines 5-7), and finally MB (Part 3, lines 8-13). In Part 2, if R is greater than the size of the current focus, then more objects (not satisfying hc) have to be added and this should be based on the approximate satisfaction of the hard constraints. This selection is done by **AppendBlocks** that is analyzed in §3.3.1. In Part 3, the block breaking for satisfying MB can be based on frequency (the user may prefer frequent or rare values and this is specified by the last parameter DV_{pref}) and it is done by **BreakBlock** that is analyzed in §3.3.2.

Algorithm 1 SmartFSRank
Input: $Obj, hc, sc, MB, R, DV_{pref}$
Output: A ranked answer that satisfies hc, sc, MB and R.

1: $A \leftarrow E(hc)$; ▷ The objects satisfying the hc
2: */** Part (1): Apply the PFS method to satisfy sc*/*
3: Compute $(A, \succ_{sc\ |A})$ which is a series of blocks $L = \langle b_1, \ldots, b_Z \rangle$.
4: */** Part (2): Satisfy R */*
5: **if** $|A| < R$ **then** ▷ need to add more objects
6: $L \leftarrow L$.**AppendBlocks**$(R - |A|)$; ▷ Add new blocks to the answer
7: **end if**
8: */** Part (3): Satisfy MB*/*
9: **for** each $b \in L$ **do**
10: **if** $|b| > MB$ **then** ▷ If block b does not satisfy MB
11: Replace b by **BreakBlock**$(b, MB, 1, DV_{pref})$
12: **end if**
13: **end for**

3.3.1 AppendBlocks

The idea is to score each object according to its distance from the hc. Having such a scoring function, a method to realize AppendBlocks is to score each object not in $E(hc)$ and return the $R - |E(hc)|$ objects that have the highest score. This method guarantees that it will return the objects which maximize the score, i.e. those that better approximate the information need, as expressed by the hc. As regards the scoring, below is detailed a method based on facet types. Let $hc = c_1 \wedge \ldots \wedge c_n$, where each conjunct c_i is a hard constraint like $F_i = t_i$, e.g Stars = 4. If o_j is an object, o_{ji} denotes the value of o_j on the facet F_i. Table 3.1 defines the score per conjunct based on the facet type. The first column corresponds to the case where a conjunct is satisfied, while the second presents the formulas used when a conjunct is not satisfied. In both cases the scores range in the interval [0, 1]. In the last row, corresponding to the case where the terminology is structured as a taxonomy, a similarity measure that reflects the distance in the taxonomy is defined. For a term $x \in F_i$, let $up(x) = \{\ t \in F_i \mid x \leq_i t\}$.

The similarity between two terms x and y, is defined as the Jaccard similarity of their greater nodes, specifically: $sim_{tax}(x,y) = \frac{|up(x) \cap up(y)|}{|up(x) \cup up(y)|}$ and then define $score_{taxonomy}(F_i = t_i, o_j) = sim_{tax}(t_i, o_{ji})$.

As an example, consider the terms of facet Location which are hierarchically organized as shown in Fig. 1.1. We can calculate similarities between terms like $sim_{tax}(Hyogo, Kyoto) = \frac{2}{4} = 0.5$ and $sim_{tax}(Hyogo, Tokyo) = \frac{1}{5} = 0.2$

Table 3.1: Formulas for calculating $score_{hc}$ per conjunct

score type	o_j **satisfies** c_i	o_j **does not satisfy** c_i
$score_{flatterminology}(F_i = t_i,\ o_j)$	1	0
$score_{numeric}(F_i = t_i,\ o_j)$	1	$1 - \frac{\lvert t_i - o_{ji}\rvert}{\max_{o_m \in Obj}\{\lvert t_i - o_{mi}\rvert\}}$
$score_{interval}(F_i \in [a,b],\ o_j)$	1	$score_{numeric}(F_i = \frac{a+b}{2},\ o_j)$
$score_{taxonomy}(F_i = t_i,\ o_j)$	$sim_{tax}(t_i, o_{ji})$	$sim_{tax}(t_i, o_{ji})$

Back to the running example of Fig. 1.2, we can now see how the scores of the approximate results were calculated, regarding the constraint Stars = 4. They both have Stars = 3, so their score [1] according to the numeric case of the above table is: $score_{numeric}(Stars = 4,\ 3) = 1 - \frac{|4-3|}{4-0} = 1 - \frac{1}{4} = 0.75$. As for the Price facet, the specified constraint is the interval [200, 2000]. Both hotels that appear in the approximate results have a price inside this interval, therefore their score for this constraint is 1. (as defined in Table 3.1, row 4, column 2). To better understand how the formula on numeric intervals works, consider the above constraint on Price and a hotel having price 100. The formula relies on the distance from the interval's center. The center is (2000 + 200)/2 = 1100, and the distance is 1100 - 100 = 1000. Then this distance is normalized by the maximum such distance in the dataset. In this dataset, for this constraint on Price facet the max distance is 1100 - 12 = 1088. Finally, the score is calculated as 1 - (1000/1088) = 1 - 0.92 = 0.08.

Definition 8 (HCscore)**.** *The consolidated score of an object o_j with respect to $hc = c_1 \wedge \ldots \wedge c_n$, can be defined as: $score_{hc}(o_j) = \frac{1}{n}\sum_{i=1}^{n} score_{type(c_i)}(o_j)$ where $type(c_i) \in \{flatterminology, numeric, interval, taxonomy\}$*

This formula can be considered as the *baseline*. It expresses how close o_j is with respect to a point that satisfies *hc*. The exact algorithm for AppendBlocks is Alg. 2. It returns the bucket order to be appended.

3.3.2 BreakBlock

For breaking each block that does not satisfy MB an idea is to score each object of the block according to its *discrimination value*. Rare elements are harder to find, therefore it could be reasonable to promote objects that have rare values, i.e. those with higher discrimination value. On the other hand, frequent

[1] Assuming that the range of the facet Stars, in the dataset, is the set {0,1,2,3,4,5}.

Algorithm 2 AppendBlocks
Input: Num
Output: A bucket order to be appended

1: $CO \leftarrow Obj \setminus E(hc)$; ▷ The candidate objects
2: **for** each $o \in CO$ **do**
3: $o.score \leftarrow score_{hc}(o)$; ▷ compute the score of o wrt hc
4: **end for**
5: $CO_{sorted} \leftarrow Sort(CO, score, descending)$; ▷ sort CO wrt $score$ attribute in desc. order
6: $i \leftarrow 0$;
7: $A \leftarrow \epsilon$;
8: **while** $|A| < Num$ **do** ▷ While the objective of Num objects is not reached
9: $A \leftarrow A \cup CO_{sorted}[i]$;
10: $i++$;
11: **end while**
12: Return A;

values may correspond to popular values, therefore it could be also reasonable to promote frequent values, i.e. those that appear in several objects. As we shall see in the section about GUI, the GUI allows the user to specify whether rare or frequent values are preferred. In any case it is necessary to define and compute the discrimination value. Such a formula can be applied to each object of any block that does not satisfy MB to order its objects. Then, such blocks will break to smaller ones satisfying MB. The *discrimination value* (dV) of an object $o_j = (o_{j1}, \ldots, o_{jk})$ can be defined by taking the average inverse frequency, i.e.: $dV_w(o_j) = \frac{1}{k} * \sum_{i=1,k} \frac{1}{freq_w(o_{ji})}$. Note that frequency can be defined in various ways, this is why the above formula uses $freq_w$ where $w \in \{g, ga, E\}$. Specifically the frequency of a value $t \in F_i$, can be defined *globally* ($freq_g$), or with respect to the objects that *have value in facet* F_i ($freq_{ga}$), or in the *current focus* E ($freq_E$). Formally: $freq_g(o_{ji}) = \frac{|\{o_x \in Obj \mid o_{xi} = o_{ji}\}|}{|Obj|}$ (1),
$freq_{ga}(o_{ji}) = \frac{|\{o_x \in Obj \mid o_{xi} = o_{ji}\}|}{|\{o_y \in Obj \mid o_{yi} \neq \epsilon\}|}$ (2), $freq_E(o_{ji}) = \frac{|\{o_x \in E \mid o_{xi} = o_{ji}\}|}{|E|}$ (3).
Note that if $o_{ji} = \epsilon$, i.e. null, then $freq_{\{g,E\}}(o_{ji})$ = number of objects having null (just like an ordinary value). Other ways to define the discrimination value of an object can be also considered. For example, in sessions where soft constraints have been set, the dV_{sc} of an object $o_j = (o_{j1}, \ldots, o_{jk})$ could be defined as the average inverse frequency in the facets used by the soft constraints sc, i.e: $dV_{w,sc}(o_j) = \frac{1}{|sc|} * \sum_{i=1,|sc|} \frac{1}{freq_w(o_{ji})}$, assuming that facets $F_1, \ldots, F_{|sc|}$ are the ones used in sc.

In general it makes sense to consider a *series of "tie breaking" methods*, for making sure that all ties can be broken so that MB is eventually satisfied. Each such method can be assigned a *level*, meaning that if the application of the level i method does not break a tie, then the level $i+1$ method is applied. The exact

Algorithm 3 BreakBlock
Input: $b, MB, level, DV_{pref}$ where b does not satisfy MB.
Output: A bucket order of the objects of b that satisfies MB.

1: $A \leftarrow objects(b)$; ▷ the objects occuring in b
2: **for** each $o \in A$ **do**
3: $o.dv \leftarrow DV(o, level)$; ▷ compute the discrimination value of o at $level$
4: **end for**
5: **if** $DV_{pref} = Rare$ **then**
6: $Sort(A, dv, descending)$; ▷ sort A wrt dv attribute in desc. order
7: **else if** $DV_{pref} = Frequent$ **then**
8: $Sort(A, dv, ascending)$; ▷ sort A wrt dv attribute in asc. order
9: **end if**
10: Let $B = \langle b_1, \ldots, b_F \rangle$ the resulting blocks ▷ after the previous sorting
11: **for** each $b \in B$ **do**
12: **if** $|b| > MB$ **then** ▷ if b_i still does not satisfy MB
13: $Bnew \leftarrow$ **BreakBlock**$(b, MB, level + 1, DV_{pref})$; ▷ recursive call with +1 level
14: **end if**
15: Replace in B the block b by the series of blocks $Bnew$
16: **end for**
17: Return B;

algorithm for BreakBlock is Alg. 3. Various series of levels can be used, that are referred to with the term *ranking policies*, such as:
$Levels_E = \langle dv_E, dv_G, lexicographic \rangle$, $Levels_G = \langle dv_G, dv_E, lexicographic \rangle$.
According to the first series, the algorithm first breaks the block b with respect to the discrimination value in the focus (i.e. $freq_E$). If MB is still not satisfied, it uses $freq_g$ (recursively only for that block). At level 3 it uses the lexicographic order with respect to the name of the object. Note that this ensures that the algorithm terminates and that it will return a bucket order that certainly satisfies MB. One key point is that a cost is paid only if needed i.e. only for the blocks that do not satisfy MB (line 12 of Alg. 3).

Chapter 4

Evaluation

Section 4.1 motivates simulation-based evaluation and provides related preliminary material. Section 4.2 presents simulation-based procedures and results related to *object ranking*, while Section 4.3 presents simulation-based procedures and results related to *approximate objects*.

4.1 Simulation-based Evaluation: Preliminaries

§4.1.1 describes the objectives and the related work of simulation-based evaluation in general, §4.1.2 describes the datasets used for carrying out the simulations, while §4.1.3 discusses two common search scenarios that will be simulated.

4.1.1 Objectives and Related Work

Objectives. The main purpose of the extended model, is to assist the user in finding the desired object. In order to evaluate the extended model with respect to that perspective, a simulation based evaluation has been conducted, since this can be *repeatable* (and thus reproducible), *more objective* and *less laborious* than an evaluation with users. In comparison to an arbitrary ranking of objects (within blocks), a good object ranking should provide an interaction experience with reduced number of hard constraints *hc*, reduced number of preferences *sc*, higher rankings of target objects, and lower navigation cost.

Previous Work on Simulation-based Evaluation. There are only a few works that evaluate faceted search through simulations. These works deal with various aspects of the interaction, including facet selection ([38], [37]), interface personalization ([16], [13]), and the Many Answers Problem ([4]). In the latter, which is more related to the topic of this thesis, they measured the cost as the expected number of queries (each on a single attribute) that need to be answered until the focus contains only the desired tuple (so no preferences are supported). This is equivalent to measuring the number of hard constraints in our evaluation setting

(if preferences are ignored). However, they focus on facet ranking, not object ranking.

4.1.2 Datasets

In these simulations 5 datasets were used. The first contains descriptions of 50 cars, described by 21 facets, the second contains information for 119 restaurants described by 22 facets, the third contains information for 382 hotels described by 18 facets, the fourth includes information for 700 fish species organized in 23 facets, and the last one is also for Fish species but has 10,000 objects. Table 4.1 summarizes their contents.

Table 4.1: The datasets used for evaluation

Dataset	**Objects**	**Facets**	**Type of contents**
D_{Cars}	50	21	Car models
D_{Rest}	119	22	Restaurants in Japan
D_{Hotels}	382	18	Hotels in Japan
$D_{Fish700}$	700	23	Fish species
$D_{Fish10K}$	10,000	23	Fish species

4.1.3 Common Scenarios

Since there are several parameters related to a simulation, a rising question is what values to use and why. For this reason, two main scenarios (or use cases) that will be used in the simulations are identified below:

- **Precision Oriented**. This scenario considers users that search for specific items. The interaction stops when the user finds the sought object. It is assumed that a user finds the sought object when it is ranked in the top 10 results. For this reason in this scenario the termination condition M (see §4.2.1) is set equal to 10. In addition, as in most search engines, the system is required to rank linearly the results, therefore MB is set to 1. This scenario is denoted by S_{Find}.

- **Recall Oriented**. This scenario assumes users that want to get information not only for one object, but for a set of objects relevant to their criteria. For this type of information need it is necessary for the system to return a minimum number of results, typically 10 or more. For this reason the parameter R is set equal to 10. This scenario is denoted by S_{Inform}.

Apart from the above cases, other cases will also be investigated (in §4.2.6).

4.2 Simulation-based Evaluation for BreakBlocks

§4.2.1 describes the simulation process and metrics, §4.2.2 the evaluated ranking methods, §4.2.3 provides the exact simulation algorithm, and §4.2.4 presents the simulation results for S_{Find}, while §4.2.5 summarizes them. More experiment results (for other values of M and MB) are given in §4.2.6. The results related to Average Rank metric are explained in §4.2.7.

4.2.1 Simulation Process and Metrics

Process. The simulated sessions correspond to users that try to find a target object by sequentially adding either hard constraints (e.g *Stars=4*) or soft constraints (e.g *prefer Stars 4 best*) that match the object's description (the conjuncts correspond to the object's facet-value pairs). The initial ordering of the objects is random, but the same in all simulated user sessions. The process terminates when the desired object is ranked in the top-M (e.g. $M = 10$) positions of the focus. During the process the metrics about the session are obtained and finally statistics are calculated. The specific measures used are described next.

Quality Metrics. The following metrics are calculated in each session:

1. *Number of constraints.* How many constraints (either hard or soft) the user needs to set in order to find the target object.

2. *Navigation cost.* The cost that a user pays trying to find the target object, that depends on the number of constraints and the cost of each constraint (this will be detailed below).

3. *Average rank.* Indicates how high or low the target object is ranked through a session. It is equal to the average value of the object's different (decreasing) ranks through a session. Note that two ranking methods in the same simulation case, could have the same number of constraints and navigation cost, however the one that ranks higher the sought objects, will get a better (lower) average rank, and thus this metric will allow us to distinguish these two ranking methods. In addition, the computation of rank at each step of the interaction should be accurate even if there are buckets, i.e. it should be computed just like the expected search length [6], e.g. if target=o_1, and $L = \langle\{o_2\}, \{o_1, o_3\}\rangle$ then the rank of o_1 is 2.5. The above calculation is denoted as $CurrentRank(L,\ o_1)$.

After all sessions for a dataset are complete, the *average* and *maximum* values of the above metrics are calculated and reported. A better ranking method should result in lower average and maximum values for all metrics.

Measuring the Navigation Cost. Let *Clicks* be the number of clicks of the simulated

user for specifying the hc (or the sc). Above it was assumed that the navigation cost, denoted by NC, is defined by $NC = |Clicks|$. However a more refined measuring is also possible, i.e.

$$NC = \sum_{c \in Clicks} cost(c) \tag{4.1}$$

where $cost(c)$ is the cost of the particular click c. The latter depends on the number of zoom-in points of that facet (the terms that appear in the left panel and the user can click for refining the focus). If only a few, then the cost is small, if many the cost is higher, e.g. it is much easier to select the correct zoom point from a list of 4 zoom points, than from a list of 15 zoom points. Based on this rationale, the formula $cost(c) = |ZoomPoints|$ can be used. Note that by measuring this cost in the simulation, it is feasible to use other formulas as well, e.g. $cost(c) = 1 + \log_{10} |ZoomPoints|$. The formula is calculated as follows: Consider a bucket order L as defined in section 3.2. Suppose that for the facet F_i, there are d in number different values among the objects in the buckets of L. In other words, in the current state of the interaction, if the user decides to form a constraint on facet F_i, then he has to select one from the d in number different values. This notation $|ZoomPoints(L, F_i)| = d$ will be used later in the simulation algorithms (specifically in Alg. 5).

4.2.2 Ranking Methods

The results from three main ranking methods will be compared:

- $\mathcal{R}_{rnd}$: No automatic ranking
- $\mathcal{R}_{rare}$: Automatic ranking based on discrimination value, preferring rare values
- $\mathcal{R}_{freq}$: Automatic ranking based on discrimination value, preferring common values

The performance of three ranking policies, which concern the last two ranking methods, will also be compared:

- $Levels_E = \langle dv_E,\ dv_G,\ lexicographic \rangle$
- $Levels_G = \langle dv_G,\ dv_E,\ lexicographic \rangle$
- $Levels_{sc} = \langle dv_{G,sc},\ dv_G,\ dv_E,\ lexicographic \rangle$

4.2.3 The General Algorithm

The exact steps of the simulation process are shown in algorithm `SimulatedUser` (Alg. 4) which in turn uses the algorithm `SimulatedSession` (Alg. 5). Alg. 4 takes a parameter M that determines the stopping condition, i.e. the interaction

stops if the sought object is in the first M positions of the focus. This parameter is in turn passed to Alg. 5. Note that each $o \in Obj$ is a candidate target. The parameter P determines the number of random permutations of facet values to consider for each object. For each object o, P sessions are simulated, by calling Alg. 5 P times with o as target. In each session all metrics are calculated. In general, the constraints applied in each session may vary (they are set on different facets). As a result, the measurements in each of the P sessions, are different. After completing the P sessions, the value of each metric is calculated as the average of the measurements taken in each individual session for the corresponding metric. Since the number of permutations may be extremely large, the iteration is done on a fixed number of random permutations (e.g $P = 10$). Notice that the paremeter P is used in lines 8-13 of Alg. 4. The parameter CT takes either the value $Hard$ or the value $Soft$ and determines if the simulated sessions will be comprised of hard or soft constraints correspondingly. Finally, the last parameter $RankMethod$ specifies which ranking method to use, i.e $\mathcal{R}_{rnd}$, $\mathcal{R}_{rare}$ or $\mathcal{R}_{freq}$.

4.2.4 Results of the scenario S_{Find}

The scenario S_{Find} was simulated on each dataset, by executing Alg. 4 with the following parameters: $MB = 1, R = 0, M = 10, P = 10$. Table 4.2 summarizes the configurations that will be evaluated making clear which cases were considered as baselines and which as methods to test [1].

Table 4.2: Baselines and methods to test in S_{Find} scenario

Parameters for Alg. 4	**Baseline/Method to test**
$CT = Hard, RankMethod = \mathcal{R}_{rnd}$	Baseline
$CT = Hard, RankMethod = \mathcal{R}_{rare}$	Method to test
$CT = Hard, RankMethod = \mathcal{R}_{freq}$	Method to test
$CT = Soft, RankMethod = \mathcal{R}_{rnd}$	Baseline
$CT = Soft, RankMethod = \mathcal{R}_{rare}$	Method to test
$CT = Soft, RankMethod = \mathcal{R}_{freq}$	Method to test

All three ranking policies were tested in this scenario. $Levels_E$ was tested for both hard constraints and soft constraints sessions. $Levels_G$ was tested only on hard constraints sessions, because in soft constraints sessions the focus does not change and as a result $dv_G(o_j) = dv_E(o_j)$ for each object $o_j \in Obj$. Finally, $Levels_{sc}$ was tested only on soft constraints sessions, since $dv_{G,sc}$ can be defined only if $sc \neq \emptyset$.

To understand the effect of automatic ranking based on the measurements, the difference between the metric results in baselines and the methods to test is calculated. This difference is expressed as a percentage to the metric value of the corresponding baseline. To make this clear, consider the following case. When

[1] In $D_{Fish10K}$, the simulation was run for 1000 randomly selected target objects, in all tests.

Algorithm 4 `SimulatedUser`
Input: $Obj, MB, R, M, P, CT, RankMethod$
Output: Mean Average Rank, Max Average Rank, Average Constraints, Max Constraints, Average Navigation Cost, Max Navigation Cost

```
/** Part (1) Initializing the variables for holding values of metrics */
avgRanks ← float array of size |Obj|;                ▷ Average rank for each object
constraints ← float array of size |Obj|;             ▷ Number of constraints for each object
navigationCosts ← float array of size |Obj|;         ▷ Navigation cost for each object
/** Part (2) Estimating quality metrics for each object in the dataset */
for each o ← (v_1, ..., v_k) ∈ Obj do
    target ← o;
    for each p = 1 ... P do
        SimulatedSession(Obj, MB, R, M, CT, RankMethod, target);
    end for
    avgRanks[target] ← avgRanks[target] / P;         ▷ Average rank for target;
    constraints[target] ← constraints[target] / P;   ▷ Average constraints for target
    navigationCosts[target] ← navigationCosts[target] / P;   ▷ Average navigation cost for target
end for
/** Part (3) Calculating metrics for the dataset */
MeanAverageRank ← average(avgRanks);
MaxAverageRank ← max(avgRanks);
AverageConstraints ← average(constraints);
MaxConstraints ← max(constraints);
AverageNavigationCost ← average(navigationCosts);
MaxNavigationCost ← max(navigationCosts);
```

tested S_{Find} in $D_{Fish700}$ with hard constraints sessions and policy $Levels_E$, the average navigation cost (calculated with the formula $cost(c) = |ZoomPoints|$) with $\mathcal{R}_{rnd}$ (baseline) was 247.57 but with $\mathcal{R}_{freq}$ (method to test), it fell to 232.63 (note that the navigation cost shows how many facet values were presented to the user in total throughout the simulated session). So this difference expressed as a percentage to the baseline is $(247.57 - 232.63)/247.57 = 14.94/247.57 = 6.03\%$. The above process was repeated for both types of sessions (hard constraints based and soft constraints based), and for each ranking policy.

Testing $Levels_E$. Table 4.3 shows the improvement of automatic ranking methods on each metric, for *hard constraints* and $Levels_E$.

We observe that $\mathcal{R}_{freq}$ performs better than $\mathcal{R}_{rare}$, in all metrics except for Mean Avg. Rank in D_{Hotels}, and Max Navigation Cost in $D_{Fish700}$. For instance,

Algorithm 5 `SimulatedSession`
Input: $Obj, MB, R, M, CT, RankMethod, target$
Output: Average Rank, Number of Constraints, Navigation Cost

1: $target = (v_1, \ldots, v_k)$
2: Get a random permutation $(a_1, \ldots, a_k)$ of the values $(v_1, \ldots, v_k)$
3: */** Part (1) Initializing a new session */*
4: $hc \leftarrow \emptyset$; ▷ The set of hard constraints for the current session
5: $sc \leftarrow \emptyset$; ▷ The set of soft constraints for the current session
6: tempRank ← CurrentRank(L, $target$); ▷ holds object rank for this session
7: tempConstraints ← 0; ▷ holds number of constraints for this session
8: tempNavigationCost ← 0; ▷ holds navigation cost for this session
9: */** Part (2) Simulating session by setting constraints sequentially */*
10: **for** each $a_i \in (a_1, \ldots, a_k)$ **do**
11: **if** $CT = Hard$ **then**
12: $hc \leftarrow hc \cup \{F_i = a_i\}$;
13: **end if**
14: **if** $CT = Soft$ **then**
15: $sc \leftarrow sc \cup \{prefer\ term\ F_i ... a_i$ *"best"*$\}$;
16: **end if**
17: $A \leftarrow E(hc)$; ▷ The objects satisfying the hc
18: Compute $(A, \succ_{sc\ |A})$ which is a series of blocks $L = \langle b_1, \ldots, b_Z \rangle$.
19: $L \leftarrow$ `SmartFSRank`$(Obj, hc, sc, MB, R, RankMethod)$;
20: tempConstraints++;
21: tempRank ← tempRank + CurrentRank(L, $target$);
22: tempNavigationCost ← tempNavigationCost + $|ZoomPoints(L,\ F_i)|$;
23: */** Part (3) Checking termination condition and obtaining metrics for this session */*
24: **if** CurrentRank(L, $target$) $\leq M$ **then**
25: tempRank ← tempRank / tempConstraints;
26: avgRanks[$target$] ← avgRanks[$target$] + tempRank;
27: constraints[$target$] ← constraints[$target$] + tempConstraints;
28: navigationCosts[$target$] ← navigationCosts[$target$] + tempNavigationCost;
29: break;
30: **end if**
31: **end for**
32: return;

Table 4.3: Improvement on S_{Find} for hard constraints sessions with policy $Levels_E$ (positive numbers indicate improvement)

Dataset	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
D_{Cars}	$\mathcal{R}_{rare}$	-3.91	-48.42	2.80	7.14	-2.62	-12.38
D_{Cars}	$\mathcal{R}_{freq}$	0.00	-25.30	2.80	7.14	-1.51	-0.65
D_{Rest}	$\mathcal{R}_{rare}$	-7.21	-89.39	0.00	-10.71	-4.17	-41.04
D_{Rest}	$\mathcal{R}_{freq}$	5.98	**36.07**	**6.35**	17.86	4.01	-21.11
D_{Hotels}	$\mathcal{R}_{rare}$	2.33	-56.02	1.50	-40.74	-3.03	-43.98
D_{Hotels}	$\mathcal{R}_{freq}$	0.19	-17.88	3.01	-29.63	-0.18	-27.83
$D_{Fish700}$	$\mathcal{R}_{rare}$	-4.69	-111.97	-1.67	0.00	-1.71	**8.76**
$D_{Fish700}$	$\mathcal{R}_{freq}$	1.20	-104.40	3.33	**30.00**	**6.03**	3.63
$D_{Fish10K}$	$\mathcal{R}_{rare}$	-8.88	-106.87	-2.80	0.00	-3.66	-12.06
$D_{Fish10K}$	$\mathcal{R}_{freq}$	**6.86**	-3.53	5.59	20.00	5.52	0.57

we can see that in D_{Rest}, Avg. Constraints are reduced by 6.3% and Avg. Navigation Cost is reduced by 4%, when using $\mathcal{R}_{freq}$.

The reason Mean Avg. Rank is not improved when enabling automatic ranking, is correlated with the decrease of Avg. Constraints and this will be explained (and supported by more experiments) later in Section 4.2.7.

Another noteworthy observation is that $\mathcal{R}_{freq}$ reduces the Max Constraints in all datasets except D_{Hotels}. This shows that $\mathcal{R}_{freq}$ reduces the hard constraints the user has to formulate not only on average, but in the worst case scenarios as well. These results indicate that for hard constraints sessions, $\mathcal{R}_{freq}$ should be preferred as a ranking method.

Table 4.4 shows the improvement of automatic ranking methods on each metric, for *soft constraints* and $Levels_E$. At first we see that $\mathcal{R}_{freq}$ improves Avg. Constraints and Max Constraints more than $\mathcal{R}_{rare}$. $\mathcal{R}_{freq}$ also improves Avg. Navigation Cost better than $\mathcal{R}_{rare}$, except for D_{Rest}. For the rest of the metrics, $\mathcal{R}_{rare}$ performs better in 3 datasets and worse in 2. From these results it is not clear which ranking method should be preferred in soft constraints sessions.

In Tables 4.3, 4.4, 4.6, 4.7 the best values in each column are highlighted, indicating the maximum gain for each metric in the simulation of each case. To see the corresponding raw measurements, the interested reader can refer to Tables B.1 - B.2 in the Appendix B.

To further summarize the results, Table 4.5 presents the average improvement of each ranking method across *all datasets*. This is illustrated more clearly in Figures 4.1 and 4.2. We see that $\mathcal{R}_{freq}$ performs clearly better in all metrics for hard constraints sessions. For soft constraints sessions, although $\mathcal{R}_{freq}$ performs slightly worse than $\mathcal{R}_{rare}$ w.r.t Mean Avg. Rank and Max Avg. Rank, it improves the other metrics more than $\mathcal{R}_{rare}$. For instance, $\mathcal{R}_{freq}$ reduces Avg. Constraints by 2.2% and Max Constraints by 16.2%.

Table 4.4: Improvement on S_{Find} for soft constraints sessions with policy $Levels_E$ (positive numbers indicate improvement)

Dataset	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
D_{Cars}	$\mathcal{R}_{rare}$	-1.66	**-1.96**	0.96	20.00	-0.65	-11.50
D_{Cars}	$\mathcal{R}_{freq}$	1.66	-7.54	0.96	20.00	0.98	6.39
D_{Rest}	$\mathcal{R}_{rare}$	-0.19	-13.95	1.69	0.00	**6.40**	16.57
D_{Rest}	$\mathcal{R}_{freq}$	**4.83**	-2.37	**3.39**	0.00	3.12	6.93
D_{Hotels}	$\mathcal{R}_{rare}$	3.61	-57.52	0.81	-11.11	-2.86	9.52
D_{Hotels}	$\mathcal{R}_{freq}$	1.80	-58.06	2.44	-5.56	2.86	4.68
$D_{Fish700}$	$\mathcal{R}_{rare}$	1.02	-79.58	0.00	0.00	0.79	-0.06
$D_{Fish700}$	$\mathcal{R}_{freq}$	0.00	-70.64	1.83	11.76	2.13	-1.38
$D_{Fish10K}$	$\mathcal{R}_{rare}$	4.29	-48.49	-0.88	0.00	-2.38	0.00
$D_{Fish10K}$	$\mathcal{R}_{freq}$	-3.05	-96.19	2.63	**55.00**	0.90	**35.36**

Table 4.5: Average performance of ranking methods on S_{Find} with policy $Levels_E$ (positive numbers signify improvement)

Constraint Type	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
Hard	$\mathcal{R}_{rare}$	-4.47	-82.53	-0.03	-8.86	-3.04	-20.14
Hard	$\mathcal{R}_{freq}$	2.84	-23.01	4.22	9.07	2.78	-9.08
Soft	$\mathcal{R}_{rare}$	1.42	-40.30	0.52	1.78	0.26	2.90
Soft	$\mathcal{R}_{freq}$	1.05	-46.96	2.25	16.24	2.00	10.40

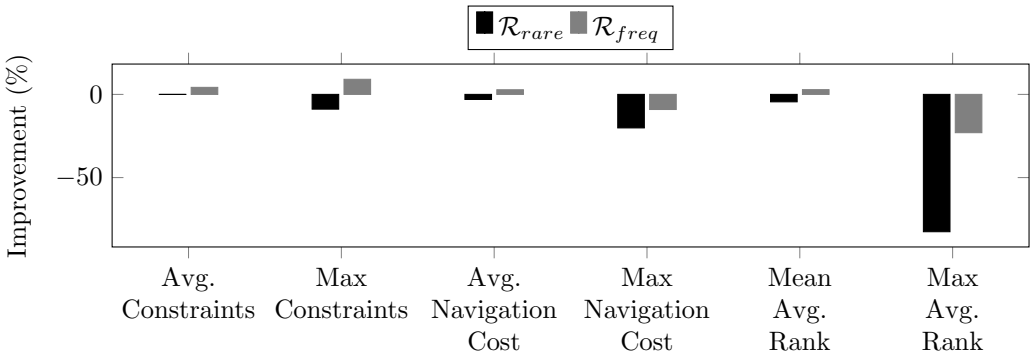

Figure 4.1: Impact of Automatic Ranking (average in all datasets, hard constraints, M=10, MB=1, policy $Levels_E$)

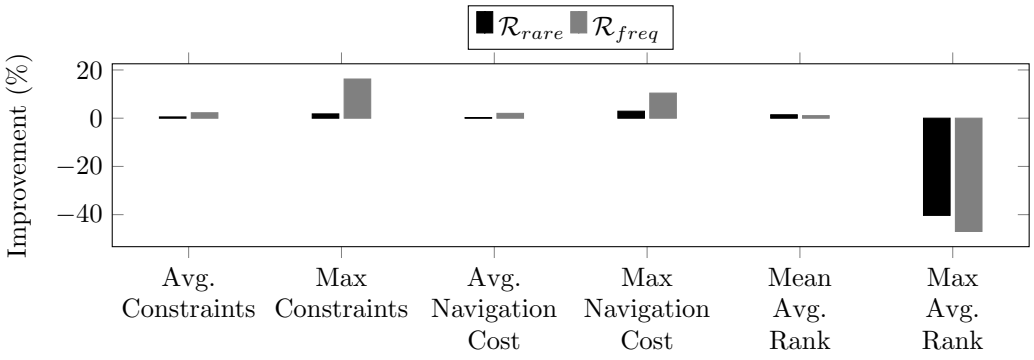

Figure 4.2: Impact of Automatic Ranking (average in all datasets, soft constraints, M=10, MB=1, policy $Levels_E$)

Table 4.6: Improvement on S_{Find} for hard constraints sessions with policy $Levels_G$ (positive numbers indicate improvement)

Dataset	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
D_{Cars}	$\mathcal{R}_{rare}$	0.38	-53.80	2.80	6.67	**11.14**	4.40
D_{Cars}	$\mathcal{R}_{freq}$	0.00	**6.57**	2.80	**20.00**	2.82	1.76
D_{Rest}	$\mathcal{R}_{rare}$	-4.08	-73.04	-4.92	-68.42	0.63	-16.60
D_{Rest}	$\mathcal{R}_{freq}$	4.79	-13.70	3.28	5.26	9.45	**13.33**
D_{Hotels}	$\mathcal{R}_{rare}$	1.69	-6.23	3.01	-29.17	4.78	-17.51
D_{Hotels}	$\mathcal{R}_{freq}$	-0.99	-69.28	5.26	-16.67	6.98	10.51
$D_{Fish700}$	$\mathcal{R}_{rare}$	-2.92	-62.49	-0.83	-41.67	-1.54	-16.89
$D_{Fish700}$	$\mathcal{R}_{freq}$	4.23	-22.93	4.96	12.50	4.42	-4.26
$D_{Fish10K}$	$\mathcal{R}_{rare}$	-3.89	-90.20	-0.69	0.00	0.48	0.00
$D_{Fish10K}$	$\mathcal{R}_{freq}$	**5.96**	5.28	**6.25**	15.00	6.56	6.29

Testing $Levels_G$. Let's now focus on $Levels_G$. Table 4.6 presents the results of S_{Find} for ranking policy $Levels_G$, for *hard constraints*. One noteworthy observation is that $\mathcal{R}_{freq}$, again (as in Tables 4.3 and 4.5) performs better than $\mathcal{R}_{rare}$ in general.

Testing $Levels_{sc}$. Table 4.7 presents the results of S_{Find} for the ranking policy $Levels_{sc}$ for *soft constraints*. We see that, and in comparison to Table 4.4, $\mathcal{R}_{freq}$ performs better than $\mathcal{R}_{rare}$ in most cases. So the policy $Levels_{sc}$ seems to improve the performance of $\mathcal{R}_{freq}$.

Comparing $Levels_E$ **vs** $Levels_G$. To compare $Levels_E$ with $Levels_G$, Table 4.8 shows the average performance of each ranking method among all datasets, for *hard constraints* sessions. At first we observe that $Levels_G$ improves the performance of $\mathcal{R}_{rare}$ in most metrics (w.r.t Avg. Constraints and Max Constraints it is slightly

Table 4.7: Improvement on S_{Find} for soft constraints sessions with policy $Levels_{sc}$ (positive numbers indicate improvement)

Dataset	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
D_{Cars}	$\mathcal{R}_{rare}$	-0.83	-11.64	1.90	14.29	4.34	4.96
D_{Cars}	$\mathcal{R}_{freq}$	0.83	-9.79	0.95	14.29	3.65	8.54
D_{Rest}	$\mathcal{R}_{rare}$	-9.20	-43.50	-3.42	-11.11	0.00	-8.20
D_{Rest}	$\mathcal{R}_{freq}$	0.96	**13.45**	**5.13**	16.67	**13.11**	3.32
D_{Hotels}	$\mathcal{R}_{rare}$	**3.18**	-42.07	0.00	-23.53	1.17	-0.57
D_{Hotels}	$\mathcal{R}_{freq}$	0.42	-36.57	3.25	-11.76	5.02	14.54
$D_{Fish700}$	$\mathcal{R}_{rare}$	-0.72	-87.10	-0.93	-6.67	-2.96	-6.12
$D_{Fish700}$	$\mathcal{R}_{freq}$	-1.37	-42.72	1.85	6.67	0.56	5.58
$D_{Fish10K}$	$\mathcal{R}_{rare}$	2.83	-43.97	0.00	0.00	-0.30	0.00
$D_{Fish10K}$	$\mathcal{R}_{freq}$	-4.83	-96.47	4.35	**52.50**	5.01	**33.64**

Table 4.8: Effects of ranking policies on S_{Find} (average results on hard constraints sessions)

Policy	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
$Levels_E$	$\mathcal{R}_{rare}$	-4.47	-82.53	-0.03	-8.86	-3.04	-20.14
$Levels_G$	$\mathcal{R}_{rare}$	-1.76	-57.15	-0.13	-26.52	3.10	-9.32
$Levels_E$	$\mathcal{R}_{freq}$	**2.84**	-23.01	4.22	**9.07**	2.78	-9.08
$Levels_G$	$\mathcal{R}_{freq}$	2.80	**-18.81**	**4.51**	7.22	**6.05**	**5.53**

decreased). Moreover, it improves $\mathcal{R}_{freq}$ in all metrics except for Avg. Constraints (see Fig. 4.3 for an illustration). Therefore, $Levels_G$ should be the preferred to $Levels_E$.

With respect to *soft constraints* sessions, $Levels_E$ and $Levels_G$ are compared in Table 4.9. It is clear from all the metrics that $Levels_{sc}$, has a negative effect on $\mathcal{R}_{rare}$. However, for $\mathcal{R}_{freq}$ we get better Avg. Constraints, Avg. Navigation Cost and Max Avg. Rank (see Fig. 4.4 for an illustration).

4.2.5 Summary of Findings (for S_{Find})

From Table 4.5 we can see that $\mathcal{R}_{freq}$ performs better than $\mathcal{R}_{rare}$ on average, for *both hard and soft constraints* sessions.

As regards the ranking policy, Table 4.8 shows that $Levels_G$ improves the performance of $\mathcal{R}_{rare}$ and $\mathcal{R}_{freq}$. Now, Table 4.9 shows that $\mathcal{R}_{rare}$ performs better in $Levels_E$, while for $\mathcal{R}_{freq}$ some metrics are improved in $Levels_{sc}$ but others are worse than in $Levels_E$. Finally, all tables show that $\mathcal{R}_{freq}$ has mostly positive

Table 4.9: Effects of ranking policies on S_{Find} (average results on soft constraints sessions)

Policy	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
$Levels_E$	$\mathcal{R}_{rare}$	**1.42**	-40.30	0.52	1.78	0.26	2.90
$Levels_{sc}$	$\mathcal{R}_{rare}$	-0.95	-45.66	-0.49	-5.40	0.45	-1.99
$Levels_E$	$\mathcal{R}_{freq}$	1.05	-46.96	2.25	**16.24**	2.00	10.40
$Levels_{sc}$	$\mathcal{R}_{freq}$	-0.80	**-34.42**	**3.11**	15.67	**5.47**	**13.12**

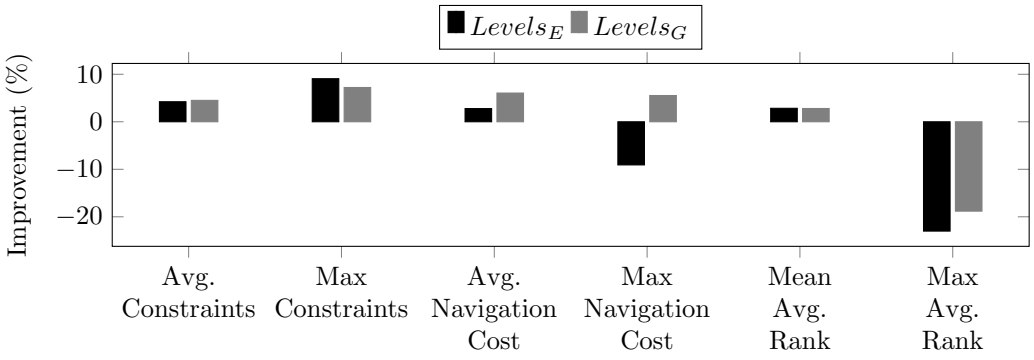

Figure 4.3: Average performance of $\mathcal{R}_{freq}$ in $Levels_E$ and $Levels_G$, for hard constraints sessions

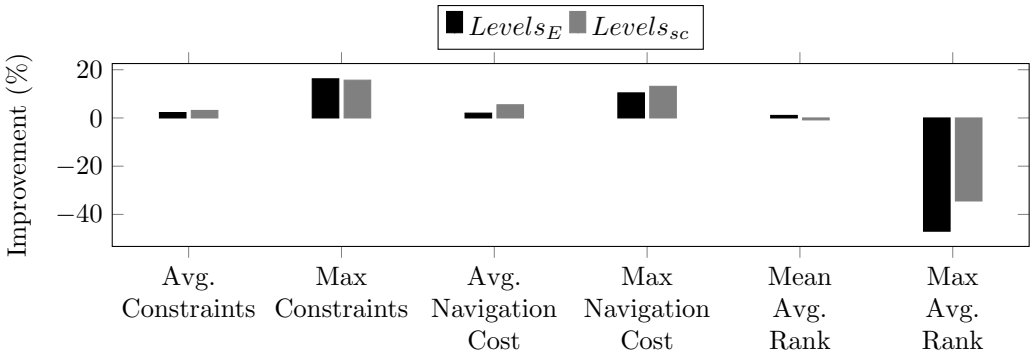

Figure 4.4: Average performance of $\mathcal{R}_{freq}$ in $Levels_E$ and $Levels_{sc}$, for soft constraints sessions

Table 4.10: Average performance of ranking methods on *hard constraints* sessions for various values of MB

MB	**Ranking Method**	**Mean Avg. Rank (%)**	**Max Avg. Rank (%)**	**Avg. Constr. (%)**	**Max Constr. (%)**	**Avg. Nav. Cost (%)**	**Max Nav. Cost (%)**
1	$\mathcal{R}_{rare}$	-5.03	-69.69	2.51	0.87	2.03	3.56
1	$\mathcal{R}_{freq}$	-1.89	**-2.55**	**9.74**	**21.80**	**9.21**	**17.65**
3	$\mathcal{R}_{rare}$	-5.53	-68.53	1.26	4.73	1.25	6.71
3	$\mathcal{R}_{freq}$	-1.01	-8.74	5.95	9.70	5.76	11.25
5	$\mathcal{R}_{rare}$	-5.17	-66.41	0.89	1.59	-0.25	3.12
5	$\mathcal{R}_{freq}$	**-0.59**	-7.29	2.92	10.92	2.18	8.11

effects on the interaction. Specifically, and with respect to the objectives stated in Section 4.1.1, $\mathcal{R}_{freq}$ with policy $Levels_E$ resulted in:

- Reduced *hc* (about 4.2% on average)
- Reduced *sc* (about 2.2% on average)
- Higher object rankings (about 2.8% and 1% on average, in hard and soft constraints sessions respectively)
- Lower navigation cost (about 2.8% and 2% on average, in hard and soft constraints sessions respectively)

In addition, we observe that $\mathcal{R}_{freq}$ reduces the *hc* and the *sc* not only on average, but for the worst cases as well, as indicated by Max Constraints and Max Navigation Cost metrics (see the last lines of Tables 4.8 and 4.9).

4.2.6 Testing Other M and MB values

In addition to S_{Find}, BreakBlocks was further evaluated by executing the algorithm for each ranking method $\mathcal{R}_{rnd}$, $\mathcal{R}_{rare}$, $\mathcal{R}_{freq}$, with three different termination conditions $M = 1, 2, 3$ and testing for $\mathcal{R}_{rare}$ and $\mathcal{R}_{freq}$ three different values of MB (1, 3, 5). The ranking policy followed in these tests is $Levels_E$.
To compare the performance of $\mathcal{R}_{rare}$ and $\mathcal{R}_{freq}$ with $MB = 1$, 3, 5, the average value of each metric across all tests ($M = 1, 2, 3$ for all datasets) is considered. Tables 4.10 and 4.11 show these average values, for hard and soft constraints sessions respectively. The raw measurements for all cases in each dataset, are available in Tables B.3 - B.7 in the Appendix B.

In Table 4.10 we observe that for hard constraints sessions, the best performing setting is $\mathcal{R}_{freq}$ with MB = 1 on all metrics except Mean Avg. Rank, where $\mathcal{R}_{freq}$ with MB = 5 performs slightly better. It should be noted that for Mean Avg. Rank and Max Avg. Rank the results are negative in all settings, but in the other metrics the results are positive.

Table 4.11: Average performance of ranking methods on *soft constraints* sessions for various values of MB

MB	Ranking Method	Mean Avg. Rank (%)	Max Avg. Rank (%)	Avg. Constr. (%)	Max Constr. (%)	Avg. Nav. Cost (%)	Max Nav. Cost (%)
1	$\mathcal{R}_{rare}$	**0.89**	-53.16	2.09	-0.44	1.96	-0.44
1	$\mathcal{R}_{freq}$	-5.53	-61.40	**7.43**	**26.84**	**7.63**	**14.74**
3	$\mathcal{R}_{rare}$	0.82	**-46.28**	1.92	-1.79	2.44	0.64
3	$\mathcal{R}_{freq}$	-5.36	-57.35	4.20	16.38	5.73	10.27
5	$\mathcal{R}_{rare}$	0.83	-50.93	0.88	1.89	1.58	0.51
5	$\mathcal{R}_{freq}$	-5.94	-67.83	2.60	10.14	3.27	6.93

In Table 4.11 we see that for soft constraints sessions, the best performing setting is $\mathcal{R}_{freq}$ with MB = 1, on Avg. Constraints, Max Constraints, Avg. Navigation Cost and Max Navigation Cost. These results provide evidence that $\mathcal{R}_{freq}$ improves the interaction. Again with respect to the objectives stated in Section 4.1.1, we find that $\mathcal{R}_{freq}$ with MB = 1 resulted in:

- Reduced *hc* (about 9.7% on average)
- Reduced *sc* (about 7.4% on average)
- Lower navigation cost (about 9% and 7.6% on average, in hard and soft constraints sessions respectively)

These results are consistent with the findings mentioned in 4.2.5.

4.2.7 Explaining Results about Average Rank

As regards the requirement about higher object rankings, we observe that $\mathcal{R}_{freq}$ (with MB = 1, 3, or 5) did not improve this aspect. Specifically, Mean Avg. Rank is increased about 1% in hard constraints sessions, and about 5.5% in soft constraints sessions. The reason Mean Avg. Rank increases when enabling automatic ranking, is correlated with the decrease of Avg. Constraints. In many cases, the session for an object with automatic ranking requires less constraints, but at the same time the Avg. Rank is increased. To make this clear, consider the following example. Assume we set M = 1, and for an object o the session without automatic ranking requires 3 constraints until rank(o) = 1. Further suppose that the rank of o after the first constraint is 10, and after the second is 2. So in this session, avg.rank(o) = (10 + 2 + 1) / 3 = 4.33. Now, suppose that in the session with $\mathcal{R}_{freq}$, only 2 constraints are required until rank(o) = 1. Also, after the first constraint the rank of o is 9. So in this case avg.rank(o) = (9 + 1) / 2 = 5, which is 15% higher than the session without automatic ranking.

4.3 Simulation-based Evaluation for AppendBlocks

Section 4.3.1 discusses the gain from AppendBlocks. Then Section 4.3.2 discusses simulation-based evaluation.

4.3.1 Benefit from AppendBlocks

AppendBlocks solves the problem of too small answers (recall scenario S_{Inform} in §4.1.3), by enriching the answer with objects that approximate the *hc*. As regards the benefits from R (and AppendBlocks), it is not hard to see that whenever a new approximate object is added, it reduces the number of constraints the user would have to formulate (for getting that object). Without AppendBlocks, the user would have to delete one filter and select another. Therefore the gain from adding $R - |A|$ objects is the number of distinct descriptions of these objects, so the gain ranges $[1, R - |A|]$.

4.3.2 Simulation-based Evaluation

One question is how to use simulation to evaluate the benefits of AppendBlocks and thus of the scoring functions that were described in §3.3.1.

Method 1. Percentage of Distinct Descriptions. One way to evaluate the benefit of AppendBlocks is to use a simplified version of Algorithm `SimulatedUser` (Alg. 4), specifically Alg. 6, that counts how many times AppendBlocks was called, and how many were the distinct descriptions of the approximate objects. For example, we can compute what percentage of the $R-|A|$ objects have distinct descriptions. If the percentage is y then we can say that the user cost is reduced by $y * (R - |A|)$ clicks.

Algorithm 6 Evaluating the Reduced Cost from R
Input: Obj, J (the number of extra approximate objects requested)
Output: Average number of less *hc* that the user has to formulate

1: **for** each $o = (v_1, \ldots, v_k) \in Obj$ or for a set of randomly selected *hc* **do**
2: Let $L = \langle b_1, \ldots, b_Z \rangle$ ▷ The produced bucket order
3: $R = |objects(L)| + J$
4: Let $L' = \langle b_1, \ldots, b_Z, b_{Z+1}, \ldots, b_{Z+V} \rangle$ ▷ The extended bucket order
5: X = Count how many new *hc* the user would have to use for getting the objects of $b_{Z+1} \ldots b_{Z+V}$, so the number of distinct *hc* (excluding $(v_1, \ldots, v_k)$)
6: $Benefit+ = X$
7: **end for**
8: return the average benefit, i.e. $Benefit/|Obj|$

Method 1: Results. To carry out these experiments Alg. 6 is used for various numbers of extra objects (i.e. for various values of J, e.g. $J = 1...10$). Specifically

Alg. 6 was run for J = 5, 10, 15, 20, 25, 30 on D_{Cars}, D_{Rest} and D_{Hotels}. For $D_{Fish700}$ the algorithm was run for J = 10, 20, 30, 40, 50. In all cases the average benefit was equal to J. That means that the simulation showed that the maximum gain was achieved in all cases.

Method 2. Coverage by the Approximate Objects. Consider a user who is interested in some ideal objects corresponding to a set of hard constraints *hc*. However suppose that no such objects exist in the dataset, therefore the user (through FS) would not even be able to express these *hc*. However, through the approximate objects of the extended model the user will be able to express these *hc* and see those objects that more closely satisfy the *hc*. This makes sense also in FS systems through spoken dialogue (e.g. see [23]) where the user is not able to see the zoom points, therefore quite often he/she expresses constraints with empty answers. Based on the above scenario, we could try measuring to what extent the approximate objects cover the *hc*.

Note that AppendBlocks does not have any explicit diversification objective. It should also be noted that a "plain vanilla" diversification that aims at covering the space (of *hc*) is not necessarily the best for the user. For example, suppose the *hc* is stars=5 and price=50, then a hotel with stars=4 and price=51 is more desired than the following pair of hotels that covers the entire *hc*: stars=5, price=100 and stars=2, price=50. However it makes sense to investigate the following research question: *to what extent the approximate objects (as defined in this thesis) cover the hc*. The Coverage by the Approximate Objects can be measured by simulating users that express their criteria through hard constraints corresponding to random facet-value pairs. Every simulated session consists of a fixed number of constraints where only one constraint is set on each facet. On such sessions we can then count how frequently the user would get an empty answer, i.e. what percentage of the simulated sessions resulted in an empty answer. For measuring the coverage of AppendBlocks, we can consider these empty answer cases, and count in how many of these cases the objects returned by AppendBlocks cover the user's information need, i.e how frequently the returned approximate objects covered (collectively) *all* the facet values specified in the hard constraints. Obviously, this depends also on the value of the R parameter, therefore it makes sense to test various values, like $R = 10, 20, 30....$ Specifically, if $Answers$ is the total number of simulated sessions, we shall denote by EA_{avg} (from Empty Answers) the average number of sessions where an empty answer was returned (there was no object satisfying all the hard constraints) i.e. $EA_{avg} = \frac{EA}{Answers}$.

We can now define $ApBl_{avg}$ as $\frac{Answers_{ApBl}}{Answers}$, where $Answers_{ApBl}$ is the number of sessions where an empty answer was returned and the approximate results covered the information need. Let $ApBl_{Cvrg,i}$ be the number of hard constraints covered by approximate results (returned by AppendBlocks) in session i, in which an empty answer was returned. Now let $ApBl_{Cvrg,avg}$ be the average number of hard constraints *covered* by approximate results, on sessions where an empty

Table 4.12: Average results of AppendBlocks evaluation

R	EA_{avg} (%)	$ApBl_{avg}$ (%)	$ApBl_{Cvrg,avg}$ (%)
10	97.47	18.61	60.84
20	97.44	24.97	65.80

answer was returned, calculated as: $ApBl_{Cvrg,avg} = \frac{1}{EA} * \sum_{i=1}^{EA} \frac{ApBl_{Cvrg,i}}{HC_{size}}$.

Note that coverage is a metric that has been also used for evaluating diversification algorithms (e.g. see [8, 2]). However, since the approach of this thesis is not intended for diversification, coverage is defined differently: as a percentage of the user's *hc* (and not w.r.t all the facets).

The algorithm that makes this simulation and computes the aforementioned metrics is `ApproximateResultsEvaluation` (Alg. 7). It takes as parameter the value of R and a parameter HC_{size} that specifies how many hard constraints will be set in each simulated session (it should not exceed the number of facets). It also takes as a parameter *Sessions* that determines how many sessions will be simulated.

Method 2: Results. The scenario S_{Inform} was tested on each dataset, by executing the simulation algorithm. The values tested for HC_{size} were 4, 5 and 6, while *Sessions* was set to 1000. The average results are shown in Table 4.12. We can see that 97.4% of the simulated sessions resulted in an empty answer. In these cases, 18.6% (for $R = 10$) and 25% (for $R = 20$) of the time AppendBlocks returned approximate results that completely covered the information need (the returned objects collectively satisfied all hard constraints). The average coverage of the information need was 60.8% and 65.8% for $R = 10$ and $R = 20$ respectively. To conclude we have seen that apart from saving user effort, AppendBlocks covers quite satisfactorily the information space. The evaluation results for each dataset, are available in Table B.8 in the Appendix B.

Algorithm 7 `ApproximateResultsEvaluation`
Input: $Obj, R, HC_{size}, Sessions$
Output: Avg. Empty Answers, Avg. AppendBlocks Answers, Avg. AppendBlocks Coverage

1: $EA \leftarrow 0$; $Answers_{ApBl} \leftarrow 0$; $ApBl_{Cvrg} \leftarrow 0$;
2: Let $F = \{F_1, \ldots, F_k\}$ be the set of facets
3: **for** $s = 1, \ldots, Sessions$ **do**
4: $hc \leftarrow \emptyset$;
5: **for** $p = 1, \ldots, HC_{size}$ **do**
6: $F_i \leftarrow$ GetRandomElement(F); ▷ Selects a random element from F
7: Let $D = \{t_1, \ldots, t_m\}$ be the domain of facet F_i
8: $t_i \leftarrow$ GetRandomElement(D);
9: $hc \leftarrow hc \cup \{F_i = t_i\}$;
10: $F \leftarrow F \setminus \{F_i\}$;
11: **end for**
12: $A \leftarrow E(hc)$; ▷ The objects satisfying the hc
13: **if** $A = \emptyset$ **then** ▷ Empty answer i.e no object satisfies all the hc
14: EA++;
15: $A \leftarrow$ AppendBlocks(R); ▷ Fills answer with R approx. objects
16: $conjunctsCovered \leftarrow 0$; ▷ Conjuncts satisfied by at least one object in approx. results
17: **for each** conjunct $\{F_i = t_i\} \in hc$ **do**
18: **if** $\exists o_j \in A$ with $o_{ji} = t_i$ **then**
19: $conjunctsCovered$++; ▷ It covers this constraint
20: **end if**
21: **end for**
22: $ApBl_{Cvrg} \leftarrow ApBl_{Cvrg}$ + ($conjunctsCovered$ / HC_{size});
23: **if** $conjunctsCovered = HC_{size}$ **then** ▷ All constraints were covered
24: $Answers_{ApBl}$++;
25: **end if**
26: **end if**
27: **end for**
28: $ApBl_{Cvrg,avg} \leftarrow ApBl_{Cvrg}$ / EA;
29: $EA_{avg} \leftarrow EA$ / $Sessions$;
30: $ApBl_{avg} \leftarrow Answers_{ApBl}$ / $Sessions$;
31: Return EA_{avg}, $ApBl_{avg}$, $ApBl_{Cvrg,avg}$;

Chapter 5

Implementation and Comparison to Related Systems

Section 5.1 discusses the implementation, Section 5.2 discusses efficiency, Section 5.3 discusses the GUI extensions that were required, and finally Section 5.4 compares the implementation of the extended model with respect to related tools/systems.

5.1 Implementation

The implementation of the proposed extended FS model is based on `Hippalus` [22], which is a publicly accessible web system that implements the PFS interaction model. The information base that feeds `Hippalus` is represented in RDF/S[1] using a schema adequate for representing objects described according to dimensions with hierarchically organized values.

The server-side of the system is implemented in Java EE v7.0 and deployed with Tomcat Server v9.0. The data management layer is based on Sesame RDF database v2.7.12 which supports RDF/S inferencing and querying. The front-end of the system is implemented using HTML/CSS/Javascript.

5.2 Efficiency

Although scalability is not the main focus, the time complexity of the algorithms is described below and time measurements are reported. The time complexity of the ranking methods is $O(N*K)$ where N, K are the number of objects and facets.

The simulation algorithms (Alg. 4, 7) were implemented as JUnit tests (v4.12) to assist modular execution. All simulations were executed using a machine with specifications that are listed in Table 5.1.

[1] http://www.w3.org/TR/rdf-schema/

As a basic time efficiency improvement, Java's *parallelStream*(), introduced in JDK 1.8, was utilized to iterate over the objects. This reduced the execution time of Alg. 4 about 40% on average, because it enabled the parallel calculation of each object's discrimination value. This improvement however, depends on the number of processor threads (cores) available.

Table 5.1: Specifications of the machine used for evaluation

Component	**Details**
CPU	Intel Core i7-3770, 3.40 GHz
Cache	L3: 8 MB, L2: 256 KB (×4)
RAM	16 GB DDR3, 1600 MHz
OS	Windows 10 Enterprise 64-bit

Table 5.2 reports the execution times of the simulations for S_{Find} with hard constraints, while Table 5.3 shows the execution times of the simulations for S_{Find} with soft constraints. The fourth column (Total Overhead) shows how much the execution time of the simulation increases when enabling automatic ranking, in comparison to the execution time of the baseline. For example in D_{Cars}, the simulation of the baseline method $\mathcal{R}_{rnd}$ with soft constraints took 3.14 seconds to complete, while $\mathcal{R}_{rare}$ needed 4.9 seconds, which is a (4.9 - 3.14)/3.14 $\approxeq$ 0.56 = 56% increase. The fifth column displays the average time per constraint in seconds, which is computed as: (execution time of simulation) / (number of constraints in the simulation). The last column (Overhead per constraints) shows the average overhead per constraint, and is calculated in the same way as the fourth column (Total Overhead), but on data from the 5th column (Avg. Constraint Time). The measurements from all datasets and more specifically the overhead per constraint measure (6th column), show that automatic ranking, as expected, slows the response time of the system. However this increase is in the same order of magnitude and ranges from 15.3% up to 141%.

In practice this delay is not noticeable in datasets D_{Cars}, D_{Rest} and D_{Hotels}, for both hard and soft constraints, since the average time per constraint is under 50 ms. For $D_{Fish700}$, the time per constraint increases from 266 ms to 575 ms (hard constraints) and from 350 ms to 609 ms (soft constraints) which is still very difficult for a user to notice. Finally, in $D_{Fish10K}$, we observe an increase from 2.27 sec to 2.8 sec (hard constraints) and from 20.28 sec to 40.36 sec (soft constraints). This is obviously noticeable, and therefore it is worth tackling in the future, e.g. by investigating the applicability of top-K algorithms, an issue that goes beyond the scope of this thesis. This concerns not only the proposed extended model but also algorithms for PFS.

Table 5.2: Execution times on S_{Find} for hard constraints sessions with policy $Levels_G$

Dataset	Ranking Method	Time in seconds	Total Overhead (%)	Avg. Time per constraint (seconds)	Overhead per constraint (%)
D_{Cars}	$\mathcal{R}_{rnd}$	4.06		0.007	
D_{Cars}	$\mathcal{R}_{rare}$	7.11	75.37	0.012	79.75
D_{Cars}	$\mathcal{R}_{freq}$	4.00	-1.45	0.006	-1.45
D_{Rest}	$\mathcal{R}_{rnd}$	23.90		0.016	
D_{Rest}	$\mathcal{R}_{rare}$	33.07	38.35	0.022	38.69
D_{Rest}	$\mathcal{R}_{freq}$	26.74	11.87	0.019	15.31
D_{Hotels}	$\mathcal{R}_{rnd}$	188.41		0.037	
D_{Hotels}	$\mathcal{R}_{rare}$	237.24	25.92	0.046	24.42
D_{Hotels}	$\mathcal{R}_{freq}$	217.48	15.43	0.044	18.65
$D_{Fish700}$	$\mathcal{R}_{rnd}$	2,241.24		0.266	
$D_{Fish700}$	$\mathcal{R}_{rare}$	4,898.06	118.54	0.575	116.03
$D_{Fish700}$	$\mathcal{R}_{freq}$	4,281.82	91.05	0.528	98.47
$D_{Fish10K}$	$\mathcal{R}_{rnd}$	32,722.50		2.272	
$D_{Fish10K}$	$\mathcal{R}_{rare}$	38,029.59	16.22	2.623	15.42
$D_{Fish10K}$	$\mathcal{R}_{freq}$	38,011.87	16.16	2.816	23.91

5.3 Extensions of the Graphical User Interface

The rising question is how to extend the GUI of a system supporting FS or PFS, to accommodate for R and MB parameters and to make clear the bucket ordering that has been used. In addition, an *explanation service* is a nice to have feature for reasons of transparency.

In general the following GUI-related questions were identified: (a) how to make evident the automatic ranking, (b) how to enable the user to change the ranking (e.g. frequent vs rare), (c) how to make clear the objects that do not satisfy the hard constraints, (d) how to provide ranking explanation (both for *hc* and *sc*).

Below is described how the above questions were tackled, by showing screenshots from the implementation. The bucket order is presented by separating buckets with a line label "Top-ranked", "Second-ranked", etc. so that the preference-based ranking is made clear to the user. The objects within a preference-based bucket are ordered based on the automatic method presented in this work (instead of an arbitrary one). This is clear in figures 1.2 and 5.2. The settings provided are the following:

Table 5.3: Execution times on S_{Find} for soft constraints sessions with policy $Levels_E$

Dataset	Ranking Method	Time in seconds	Total Overhead (%)	Avg. Time per constraint (seconds)	Overhead per constraint (%)
D_{Cars}	$\mathcal{R}_{rnd}$	3.14		0.006	
D_{Cars}	$\mathcal{R}_{rare}$	4.90	56.38	0.009	56.38
D_{Cars}	$\mathcal{R}_{freq}$	3.08	-1.91	0.006	-1.18
D_{Rest}	$\mathcal{R}_{rnd}$	14.27		0.009	
D_{Rest}	$\mathcal{R}_{rare}$	26.75	87.43	0.017	90.63
D_{Rest}	$\mathcal{R}_{freq}$	17.93	25.60	0.012	38.01
D_{Hotels}	$\mathcal{R}_{rnd}$	132.49		0.024	
D_{Hotels}	$\mathcal{R}_{rare}$	312.78	136.09	0.057	141.38
D_{Hotels}	$\mathcal{R}_{freq}$	266.45	101.11	0.049	108.09
$D_{Fish700}$	$\mathcal{R}_{rnd}$	2,661.13		0.350	
$D_{Fish700}$	$\mathcal{R}_{rare}$	4,646.21	74.60	0.609	74.11
$D_{Fish700}$	$\mathcal{R}_{freq}$	4,022.14	51.14	0.539	53.98
$D_{Fish10K}$	$\mathcal{R}_{rnd}$	231,855.69		20.285	
$D_{Fish10K}$	$\mathcal{R}_{rare}$	464,159.81	100.19	40.362	98.97
$D_{Fish10K}$	$\mathcal{R}_{freq}$	420,386.88	81.31	37.975	87.21

1. Enable / disable $Rsat$
2. Specify the value of R parameter
3. Enable / disable inner bucket ordering
4. Enable / disable $MBsat$
5. Specify the value of MB parameter
6. Select policy about discrimination value: prefer rare values, common values or no preference

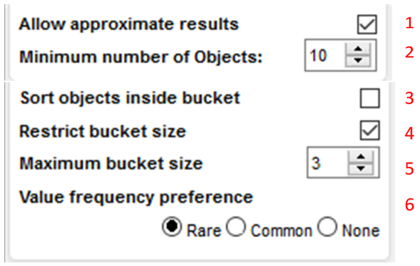

Figure 5.1: The automatic ranking settings as provided in the GUI

Figure 5.1 shows the settings used in the running example (Fig. 1.2).

As regards *rank explanation*, for each object of the focus the extended GUI shows its score as a percentage (consolidated $score_{hc}, dv_E$, and dv_G) as shown in the right side of Fig. 1.2. Moreover, for each object the GUI provides a button labeled "explain" that when pressed, it displays the $score_{hc}$ per facet, as well as the soft constraints and which of them are satisfied by the object. As an example, Figure 5.3 shows the "explanation card" for one hotel that belongs to the approximate results of our running example (Fig. 1.2), and this is why its

background color is red.

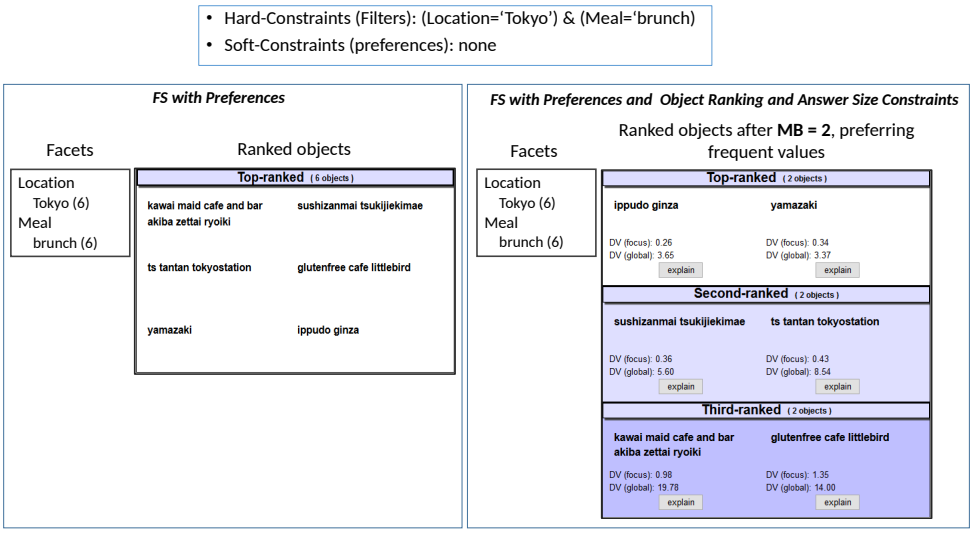

Figure 5.2: An example where a user searches for restaurants in Tokyo that offer brunch. The left side shows a typical FS response with 2 filters, while the right side shows the response of the extended FS which provides a *more refined ranking*.

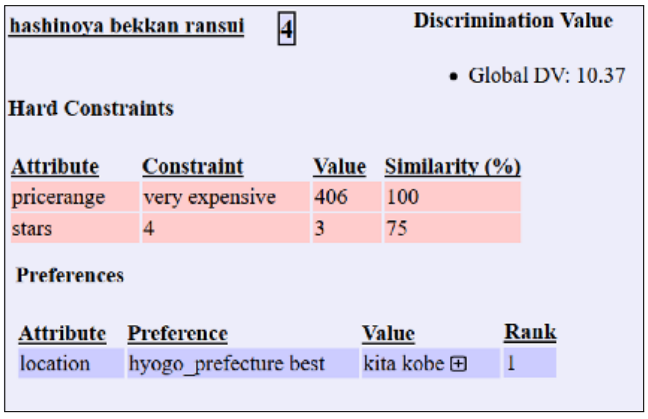

Figure 5.3: Information about the scores of an object

The above GUI is indicative and is given only for showing how a user could get the explanation if they wish too. A simple GUI that implicitly assumes $MB = 1$ and R equal to the number that determines the pagination of the results, is also a means to exploit the results of the extended model without having to add anything to the current GUIs.

5.4 Comparison with Related Systems

The research prototype system derived by enriching `Hippalus` with the functionality described in this thesis is called $\texttt{Hippalus}_{MB,R}$. By comparing $\texttt{Hippalus}_{MB,R}$ with other related tools we could say that this is the first system that supports hard constraints (the typical functionality of FS), soft constraints (including preference inheritance in the hierarchically organized values), object ranking that considers the soft constraints as well as the frequency of the data values, and supports the answer size and object granularity constraints. Since object ranking does not presuppose the existence of log files or training data, it can be widely applied on any dataset that describes objects according to a multi-dimensional space.

There is no other directly related system to compare with. The closer works to this thesis, are [4], [36] and [20]. In comparison to [4], this work focuses on object ranking while they focus on facet ranking. However, their evaluation is also done by simulations, and one of the metrics used is the number of constraints, one metric that is also used in the simulations of this thesis. In comparison to [36], that work relies on machine learning techniques and user data; while this work focuses on the query constraints and the statistical properties of the dataset. Finally, [20] focuses on open domain FS over RDF data, it assumes that entities are a result of a keyword search query, and the ranking of objects is based on text similarity functions as well as the properties of the knowledge graph. Our work on the other hand, does not require any keyword search query; instead the ranking is based on the user actions.

To make clear the key differences between $\texttt{Hippalus}_{MB,R}$ and the most related systems (that were mentioned in §2.2), Table 5.4 provides a list of features and marks those systems that provide them.

Table 5.4: Comparison with Related Systems

	$\texttt{Hippalus}_{MB,R}$	GRAFA [20]	Basu et al. [4]	van Belle [36]
Data Representation				
Facet Hierarchies	Yes	No	Yes	No
Multi-valued Facets	Yes	Yes	No	Yes
Availability of External/Log Data				
Needs training data	No	No	No	Yes
Result set Characteristics				
Approximate Matching	Yes	No	No	Yes
Blocks of desired size	Yes	No	No	No
Preferences	Yes	No	No	No

Chapter 6

Conclusion

Although Faceted Search systems are widely used, the issue of object ranking is surprisingly not well elaborated. In this thesis, an extended model for FS is proposed that aims at improving the exploration experience of the users. Specifically, two parameters are proposed that specify the desired properties of the returned answers (in terms of size and ranking granularity). Subsequently, and through the algorithm `SmartFSRank` the problem was factorized to sub-tasks that can be tackled more easily. It was shown that by using this algorithm, which takes into account (a) hard-constraints, (b) soft-constraints (preferences), and (c) the statistical properties of the dataset, an object ranking can be produced. Then the resulting ranking was evaluated through simulation, for testing whether it reduces the user effort and improves the answers. Various cases were comparatively evaluated. The results provide evidence that the proposed model reduces the user's cost for finding the desired object, and improves the interaction experience. Specifically we have seen that the intra-block ranking method $\mathcal{R}_{freq}$ with $MB = 1$, is beneficial in most cases regarding both the average cost (9.7% and 7.4%) and the maximum cost (21.8% and 26.8%), for hard and soft constraints respectively.

As regards the *approximate* objects returned by AppendBlocks, they alleviate the user from having to change the hard constraints (and note that each approximate object saves at least one click effort from the user). Although the main objective is to return objects that better approximate the hard constraints provided by the user, the simulation-based evaluation has shown that the approximate objects returned provide a good *coverage* of all the constraints (60.8% of the user's *hard constraints* on average, when the faceted search system would not allow the formulation of all the *hard constraints* since they would lead to empty results).

Apart from the above, this work described an implementation of the model and the required GUI extensions, while paying attention also to issues of transparency and explainability.

6.1 Future Work

There are several directions and issues for further research. In the task of finding the closest approximate objects, it is worth studying query relaxation methods to reduce the number of candidate objects before applying the scoring formulas. In this case however, the system might not return the objects that have the maximum score. An interesting extension of the ranking algorithm would be to consider also diversification requirements (e.g. see [2] for a survey).

Another direction is to investigate whether the proposed simulation framework and metrics can be exploited for evaluating in a cost-effective method the effectiveness of an information access interface (avg, max cost) allowing in this way the designers to examine adding/changing facets and terms for improving the overall findability of the objects. Another issue is to investigate indexes and algorithms for *scalability* i.e. for enabling FS with automated ranking over very big datasets.

Bibliography

[1] Tim Soltvedt Aadland. Evaluating a faceted search approach for efficient news event filtering, 2020.

[2] Adnan Abid, Naveed Hussain, Kamran Abid, Farooq Ahmad, Muhammad Shoaib Farooq, Uzma Farooq, Sher Afzal Khan, Yaser Daanial Khan, Muhammad Azhar Naeem, and Nabeel Sabir. A survey on search results diversification techniques. *Neural Computing and Applications*, 27(5):1207–1229, Jul 2016.

[3] Sanjay Agrawal, Surajit Chaudhuri, Gautam Das, and Aristides Gionis. Automated ranking of database query results. In *Conference on Innovative Data Systems Research 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 5-8, 2003, Online Proceedings*, 2003.

[4] Senjuti Basu Roy, Haidong Wang, Gautam Das, Ullas Nambiar, and Mukesh Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM conference on Information and Knowledge Management*, pages 13–22. ACM, 2008.

[5] Surajit Chaudhuri, Gautam Das, Vagelis Hristidis, and Gerhard Weikum. Probabilistic ranking of database query results. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 888–899. VLDB Endowment, 2004.

[6] William S. Cooper. Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.

[7] Wisam Dakka, Panagiotis G. Ipeirotis, and Kenneth R. Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, CIKM '05, pages 768–775, New York, NY, USA, 2005. ACM.

[8] Marina Drosou and Evaggelia Pitoura. Search result diversification. *SIGMOD record*, 39(1):41–47, 2010.

[9] Leila Feddoul, Sirko Schindler, and Frank Löffler. Automatic facet generation and selection over knowledge graphs. In *International Conference on Semantic Systems*, pages 310–325. Springer, 2019.

[10] Sébastien Ferré. Sparklis: a SPARQL Endpoint Explorer for Expressive Question Answering. In *ISWC Posters & Demonstrations Track*, Riva del Garda, Italy, October 2014.

[11] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In Witold Abramowicz and Robert Tolksdorf, editors, *Business Information Systems*, pages 1–11, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[12] Andreas Harth. Visinav: Visual web data search and navigation. In Sourav S. Bhowmick, Josef Küng, and Roland Wagner, editors, *Database and Expert Systems Applications*, pages 214–228, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] Frank Hopfgartner, Thierry Urruty, Pablo Bermejo Lopez, Robert Villa, and Joemon M. Jose. Simulated evaluation of faceted browsing based on feature selection. *Multimedia Tools and Applications*, 47(3):631–662, May 2010.

[14] Ioannis Kitsos, Kostas Magoutis, and Yannis Tzitzikas. Scalable entity-based summarization of web search results using mapreduce. *Distributed and Parallel Databases*, 32(3):405–446, 2014.

[15] Yuta Kobayashi, Hiroyuki Shindo, and Yuji Matsumoto. Scientific article search system based on discourse facet representation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9859–9860, 07 2019.

[16] Jonathan Koren, Yi Zhang, and Xue Liu. Personalized interactive faceted search. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 477–486, New York, NY, USA, 2008. ACM.

[17] Chengkai Li, Ning Yan, Senjuti B. Roy, Lekhendro Lisham, and Gautam Das. Facetedpedia: Dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 651–660, New York, NY, USA, 2010. ACM.

[18] Panagiotis Lionakis and Yannis Tzitzikas. Pfsgeo: Preference-enriched faceted search for geographical data. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 125–143. Springer, 2017.

[19] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[20] José Moreno-Vega and Aidan Hogan. Grafa: Scalable faceted browsing for RDF graphs. In *International Semantic Web Conference*, pages 301–317. Springer, 2018.

[21] Eyal Oren, Renaud Delbru, and Stefan Decker. Extending faceted navigation for RDF data. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora M. Aroyo, editors, *The Semantic Web - ISWC 2006*, pages 559–572, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[22] Panagiotis Papadakos and Yannis Tzitzikas. Hippalus: Preference-enriched faceted exploration. In *EDBT/ICDT Workshops*, volume 172, 2014.

[23] Alexandros Papangelis, Panagiotis Papadakos, , Yannis Stylianou, and Yannis Tzitzikas. Spoken dialogue for information navigation. In *Special Interest Group on Discourse and Dialogue*, 2018.

[24] Olivier Pivert, Olfa Slama, and Virginie Thion. SPARQL Extensions with Preferences: a Survey. In *ACM Symposium on Applied Computing*, Pisa, Italy, France, April 2016.

[25] J Rosati, T Di Noia, R De Leone, T Lukasiewicz, and VW Anelli. Combining rdf and sparql with cp-theories to reason about preferences in a linked data setting. *Semantic Web*, 2018.

[26] Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic taxonomies and faceted search: theory, practice, and experience*, volume 25. Springer Science & Business Media, 2009.

[27] Olfa Slama. Personalized queries under a generalized user profile model based on fuzzy sparql preferences. In *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE, 2019.

[28] Kostas Stefanidis, Georgia Koutrika, and Evaggelia Pitoura. A survey on representation, composition and application of preferences in database systems. *ACM Transactions on Database Systems (TODS)*, 36(3):19, 2011.

[29] Bogaard Tessel. Metadata categorization for identifying search patterns in a digital library. 75(2):270–286, Jan 2019.

[30] Antonis Troumpoukis, Stasinos Konstantopoulos, and Angelos Charalambidis. An Extension of SPARQL for Expressing Qualitative Preferences. In *The Semantic Web – ISWC 2017*, pages 711–727, Cham, 2017. Springer International Publishing.

[31] Daniel Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–80, 2009.

[32] Yannis Tzitzikas, Nicolas Bailly, Panagiotis Papadakos, Nikos Minadakis, and George Nikitakis. Using preference-enriched faceted search for species identification. *International Journal of Metadata, Semantics and Ontologies*, 11(3):165–179, 2016.

[33] Yannis Tzitzikas and Eleftherios Dimitrakis. Preference-enriched faceted search for voting aid applications. *IEEE Transactions on Emerging Topics in Computing*, 7:218–229, 2019.

[34] Yannis Tzitzikas, Nikos Manolis, and Panagiotis Papadakos. Faceted Exploration of RDF/S Datasets: A Survey. *Intelligent Information Systems*, 48(2):329–364, April 2017.

[35] Yannis Tzitzikas and Panagiotis Papadakos. Interactive exploration of multidimensional and hierarchical information spaces with real-time preference elicitation. *Fundamenta Informaticae*, 20:1–42, 2012.

[36] Agnes van Belle. Learning to rank for faceted search: Bridging the gap between theory and practice, 2017.

[37] Damir Vandic, Steven Aanen, Flavius Frasincar, and Uzay Kaymak. Dynamic facet ordering for faceted product search engines. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1004–1016, 2017.

[38] Damir Vandic, Flavious Frasincar, and Uzay Kaymak. Facet selection algorithms for web product search. In *Proceedings of the 22nd International Conference on Information and Knowledge Management*. ACM, 2013.

Appendix A

Notations

Table A.1: Summary of notations used in Sections 3.1 and 3.2

Notation / Abbreviation	**Definition / Explanation**
$F_1, \ldots, F_k$	The facets (also known as attributes)
a_i	A user action which is either a hard constraint (restriction/filter) or a soft constraint (preference)
us	The user session which is a series of user actions, i.e. $us = \langle a_1, \ldots, a_n \rangle$
hc	The set of hard constraints (restrictions/filters) that the user has specified.
sc	The set of soft constraints (preferences) that the user has specified.
Obj	The set of objects in the dataset (a.k.a instances or tuples)
$E(hc)$	The focus, i.e. the set of objects that satisfy the set of hard constraints hc. It is a subset of Obj.
$\succ_{sc}$	The preference relation induced by sc.
$\succ_{sc \mid E(hc)}$	The restriction of the binary relation $\succ_{sc}$ over $E(hc)$, that is $\succ_{sc \mid E(hc)} = \{(a, b) \in \succ_{sc} \mid a \in E(hc), b \in E(hc)\}$
$\langle b_1, \ldots, b_z \rangle$	An ordered sequence of blocks (bucket order). Blocks are pairwise disjoint ($b_i \cap b_j = \emptyset$)
L	The system's answer which is a bucket order. It may contain more or less objects than the focus.
$objects(L)$	The set of objects in the system's answer (those returned by the system)
MB	This is a parameter of type positive integer, that specifies the maximum block size of the system's answer $L = \langle b_1, \ldots, b_z \rangle$. Each block should not contain more than MB in number objects, i.e. $\lvert b_i \rvert \leq MB, for\ each\ i = 1, \ldots, z$
R	This is a parameter of type positive integer, that specifies the exact number of objects that the system's answer should contain, i.e $\lvert objects(L) \rvert = R$

Appendix B

Evaluation measurements

Table B.1: Measurements on S_{Find} for hard constraints sessions with policies $Levels_E$ and $Levels_G$

Dataset	**Ranking Method**	**Mean Avg. Rank**		**Max Avg. Rank**		**Avg. Constr.**		**Max Constr.**		**Avg. Nav. Cost**		**Max Nav. Cost**	
		$Levels_E$	$Levels_G$	$Levels_E$	$Levels_G$	$Levels_E$	$Levels_G$	$Levels_E$	$Levels_G$	$Levels_E$	$Levels_G$	$Levels_E$	$Levels_G$
D_{Cars}	$\mathcal{R}_{rnd}$	2.56	2.60	4.11	4.87	1.07	1.07	1.40	1.50	22.54	23.78	30.70	34.10
D_{Cars}	$\mathcal{R}_{rare}$	2.66	2.59	6.10	7.49	1.04	1.04	1.30	1.40	23.13	21.13	34.50	32.60
D_{Cars}	$\mathcal{R}_{freq}$	2.56	2.60	5.15	4.55	1.04	1.04	1.30	1.20	22.88	23.11	30.90	33.50
D_{Rest}	$\mathcal{R}_{rnd}$	5.69	5.64	16.30	11.61	1.26	1.22	2.80	1.90	32.38	33.12	59.70	76.50
D_{Rest}	$\mathcal{R}_{rare}$	6.10	5.87	30.87	20.09	1.26	1.28	3.10	3.20	33.73	32.91	84.20	89.20
D_{Rest}	$\mathcal{R}_{freq}$	5.35	5.37	10.42	13.20	1.18	1.18	2.30	1.80	31.08	29.99	72.30	66.30
D_{Hotels}	$\mathcal{R}_{rnd}$	10.29	10.07	29.47	40.27	1.33	1.33	2.70	2.40	73.63	76.52	187.60	202.70
D_{Hotels}	$\mathcal{R}_{rare}$	10.05	9.90	45.98	42.78	1.31	1.29	3.80	3.10	75.86	72.86	270.10	238.20
D_{Hotels}	$\mathcal{R}_{freq}$	10.27	10.17	34.74	68.17	1.29	1.26	3.50	2.80	73.76	71.18	239.80	181.40
$D_{Fish700}$	$\mathcal{R}_{rnd}$	20.06	20.56	54.72	68.52	1.20	1.21	3.00	2.40	247.57	246.12	705.60	601.10
$D_{Fish700}$	$\mathcal{R}_{rare}$	21.00	21.16	115.99	111.34	1.22	1.22	3.00	3.40	251.80	249.90	643.80	702.60
$D_{Fish700}$	$\mathcal{R}_{freq}$	19.82	19.69	111.85	84.23	1.16	1.15	2.10	2.10	232.63	235.24	680.00	626.70
$D_{Fish10K}$	$\mathcal{R}_{rnd}$	271.70	272.48	1538.60	1537.70	1.43	1.44	4.00	4.00	1353.03	1369.01	3831.00	3831.00
$D_{Fish10K}$	$\mathcal{R}_{rare}$	295.82	283.07	3182.93	2924.73	1.47	1.45	4.00	4.00	1402.61	1362.46	4293.00	3831.00
$D_{Fish10K}$	$\mathcal{R}_{freq}$	253.07	256.25	1592.95	1456.57	1.35	1.35	3.20	3.40	1278.39	1279.20	3809.00	3589.90

Table B.2: Measurements on S_{Find} for soft constraints sessions with policies $Levels_E$ and $Levels_{sc}$

Dataset	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
		$Levels_E$	$Levels_{sc}$	$Levels_E$	$Levels_{sc}$	$Levels_E$	$Levels_{sc}$	$Levels_E$	$Levels_{sc}$	$Levels_E$	$Levels_{sc}$	$Levels_E$	$Levels_{sc}$
D_{Cars}	$\mathcal{R}_{rnd}$	2.41	2.42	3.58	3.78	1.04	1.05	1.50	1.40	21.43	23.04	31.30	36.30
D_{Cars}	$\mathcal{R}_{rare}$	2.45	2.44	3.65	4.22	1.03	1.03	1.20	1.20	21.57	22.04	34.90	34.50
D_{Cars}	$\mathcal{R}_{freq}$	2.37	2.40	3.85	4.15	1.03	1.04	1.20	1.20	21.22	22.20	29.30	33.20
D_{Rest}	$\mathcal{R}_{rnd}$	5.38	5.22	11.83	11.15	1.18	1.17	1.70	1.80	31.08	32.96	66.40	57.30
D_{Rest}	$\mathcal{R}_{rare}$	5.39	5.70	13.48	16.00	1.16	1.21	1.70	2.00	29.09	32.96	55.40	62.00
D_{Rest}	$\mathcal{R}_{freq}$	5.12	5.17	12.11	9.65	1.14	1.11	1.70	1.50	30.11	28.64	61.80	55.40
D_{Hotels}	$\mathcal{R}_{rnd}$	9.42	9.44	14.76	15.64	1.23	1.23	1.80	1.70	69.00	71.09	200.70	176.10
D_{Hotels}	$\mathcal{R}_{rare}$	9.08	9.14	23.25	22.22	1.22	1.23	2.00	2.10	70.97	70.26	181.60	177.10
D_{Hotels}	$\mathcal{R}_{freq}$	9.25	9.40	23.33	21.36	1.20	1.19	1.90	1.90	67.03	67.52	191.30	150.50
$D_{Fish700}$	$\mathcal{R}_{rnd}$	16.59	16.75	25.07	25.28	1.09	1.08	1.70	1.50	219.81	213.97	465.10	459.00
$D_{Fish700}$	$\mathcal{R}_{rare}$	16.42	16.87	45.02	47.30	1.09	1.09	1.70	1.60	218.07	220.31	465.40	487.10
$D_{Fish700}$	$\mathcal{R}_{freq}$	16.59	16.98	42.78	36.08	1.07	1.06	1.50	1.40	215.13	212.78	471.50	433.40
$D_{Fish10K}$	$\mathcal{R}_{rnd}$	190.07	190.38	336.00	337.98	1.14	1.15	4.00	4.00	1077.15	1106.33	3831.00	3831.00
$D_{Fish10K}$	$\mathcal{R}_{rare}$	181.92	184.99	498.93	486.58	1.15	1.15	4.00	4.00	1102.80	1109.67	3831.00	3831.00
$D_{Fish10K}$	$\mathcal{R}_{freq}$	195.87	199.57	659.20	664.02	1.11	1.10	1.80	1.90	1067.49	1050.87	2476.50	2542.10

Table B.3: Measurements on D_{Cars} for both hard and soft constraints sessions

M	MB	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
			hc	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*
1	-	$\mathcal{R}_{rnd}$	2.14	2.05	3.24	2.57	1.22	1.13	2.40	1.60	24.70	23.58	42.10	29.70
1	1	$\mathcal{R}_{rare}$	2.19	2.08	4.83	3.95	1.11	1.09	1.80	1.40	23.57	22.42	38.20	39.50
1	1	$\mathcal{R}_{freq}$	2.34	2.12	4.45	3.45	1.13	1.11	1.50	1.70	23.74	22.48	34.30	35.00
1	3	$\mathcal{R}_{rare}$	2.26	2.06	5.34	3.47	1.15	1.11	1.70	1.50	24.82	23.88	39.00	33.90
1	3	$\mathcal{R}_{freq}$	2.02	2.08	3.65	3.43	1.15	1.12	1.90	1.70	24.01	23.32	39.80	36.00
1	5	$\mathcal{R}_{rare}$	2.07	2.06	3.41	4.12	1.18	1.15	1.80	1.70	26.41	23.92	43.90	37.30
1	5	$\mathcal{R}_{freq}$	2.18	2.06	4.92	4.50	1.12	1.13	1.40	1.60	25.76	23.91	37.20	35.00
2	-	$\mathcal{R}_{rnd}$	2.37	2.26	3.16	3.02	1.09	1.08	1.50	1.60	23.21	23.49	34.00	34.40
2	1	$\mathcal{R}_{rare}$	2.29	2.28	5.44	4.43	1.07	1.05	1.50	1.30	22.27	22.85	36.60	30.30
2	1	$\mathcal{R}_{freq}$	2.33	2.23	5.00	3.80	1.11	1.06	1.50	1.40	24.05	23.36	33.90	37.40
2	3	$\mathcal{R}_{rare}$	2.39	2.34	4.29	4.05	1.07	1.04	1.30	1.30	22.65	20.96	31.90	31.60
2	3	$\mathcal{R}_{freq}$	2.34	2.32	3.79	3.77	1.07	1.04	1.70	1.50	21.61	21.80	34.60	31.70
2	5	$\mathcal{R}_{rare}$	2.43	2.22	4.92	4.15	1.09	1.09	1.40	1.40	23.46	23.28	35.50	34.80
2	5	$\mathcal{R}_{freq}$	2.34	2.35	4.18	4.01	1.11	1.06	1.60	1.50	22.70	21.47	34.20	30.90
3	-	$\mathcal{R}_{rnd}$	2.39	2.34	3.55	2.90	1.09	1.07	1.30	1.30	23.23	22.27	33.70	32.80
3	1	$\mathcal{R}_{rare}$	2.35	2.34	5.65	4.70	1.06	1.03	1.30	1.20	23.36	22.46	34.10	29.80
3	1	$\mathcal{R}_{freq}$	2.39	2.31	4.40	3.78	1.07	1.06	1.30	1.40	23.55	21.93	33.90	31.80
3	3	$\mathcal{R}_{rare}$	2.40	2.33	5.17	4.16	1.09	1.03	1.40	1.30	24.01	21.79	32.60	33.10
3	3	$\mathcal{R}_{freq}$	2.51	2.31	5.47	4.08	1.10	1.07	1.50	1.40	22.68	22.77	32.60	34.90
3	5	$\mathcal{R}_{rare}$	2.47	2.42	3.95	4.28	1.06	1.04	1.40	1.20	22.31	20.78	31.30	29.20
3	5	$\mathcal{R}_{freq}$	2.58	2.36	7.28	3.62	1.08	1.04	1.50	1.40	22.76	21.39	33.70	31.80

Table B.4: Measurements on D_{Rest} for both hard and soft constraints sessions

M	MB	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
			hc	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*
1	-	$\mathcal{R}_{rnd}$	3.18	2.86	16.72	4.43	2.13	1.68	12.00	12.00	55.74	42.93	282.00	282.00
1	1	$\mathcal{R}_{rare}$	3.50	2.94	35.68	6.32	1.92	1.58	8.00	12.00	50.24	42.19	174.00	282.00
1	1	$\mathcal{R}_{freq}$	3.33	3.13	9.50	7.83	1.65	1.41	12.00	2.10	42.14	37.16	282.00	72.90
1	3	$\mathcal{R}_{rare}$	3.36	2.76	29.88	6.23	2.18	1.63	12.00	12.00	56.66	43.01	282.00	282.00
1	3	$\mathcal{R}_{freq}$	2.98	2.72	8.07	6.85	2.16	1.68	12.00	12.00	55.99	41.79	282.00	282.00
1	5	$\mathcal{R}_{rare}$	3.24	2.78	31.67	7.50	2.18	1.66	12.00	12.00	56.70	43.74	282.00	282.00
1	5	$\mathcal{R}_{freq}$	3.03	2.94	8.04	8.36	2.12	1.67	12.00	12.00	54.20	43.47	282.00	282.00
2	-	$\mathcal{R}_{rnd}$	3.79	3.39	14.57	5.38	1.59	1.40	5.40	2.10	42.50	38.89	119.60	73.30
2	1	$\mathcal{R}_{rare}$	3.69	3.47	15.05	7.83	1.57	1.36	5.10	2.10	40.41	37.35	140.10	68.70
2	1	$\mathcal{R}_{freq}$	3.80	3.52	8.96	7.31	1.46	1.32	3.60	2.10	38.45	35.40	86.90	74.30
2	3	$\mathcal{R}_{rare}$	4.23	3.62	19.69	9.78	1.50	1.32	3.60	2.00	36.51	34.85	82.90	59.50
2	3	$\mathcal{R}_{freq}$	4.08	3.86	10.43	7.38	1.33	1.25	2.50	1.80	35.65	31.51	83.50	61.10
2	5	$\mathcal{R}_{rare}$	3.77	3.27	16.07	7.78	1.68	1.40	5.00	2.10	44.21	36.46	106.00	79.10
2	5	$\mathcal{R}_{freq}$	3.61	3.57	8.73	9.01	1.63	1.41	5.20	2.10	42.39	38.37	119.20	83.20
3	-	$\mathcal{R}_{rnd}$	4.07	3.71	18.75	5.77	1.48	1.34	6.50	1.80	38.05	36.39	153.50	68.90
3	1	$\mathcal{R}_{rare}$	4.45	3.63	28.65	7.29	1.53	1.31	5.60	1.90	39.14	34.63	122.00	64.70
3	1	$\mathcal{R}_{freq}$	4.06	3.58	9.38	7.52	1.35	1.25	2.50	2.10	36.19	33.38	72.00	70.10
3	3	$\mathcal{R}_{rare}$	4.10	3.59	22.65	8.02	1.56	1.33	5.00	2.10	39.76	33.46	105.90	69.40
3	3	$\mathcal{R}_{freq}$	4.14	3.95	9.91	7.68	1.39	1.25	4.40	2.10	37.68	32.77	131.20	66.00
3	5	$\mathcal{R}_{rare}$	4.64	4.09	27.22	8.08	1.41	1.26	5.40	1.90	37.84	34.18	119.90	65.60
3	5	$\mathcal{R}_{freq}$	4.59	4.35	11.07	8.69	1.30	1.20	3.60	1.70	34.93	30.49	97.10	64.50

Table B.5: Measurements on D_{Hotels} for both hard and soft constraints sessions

M	MB	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
			hc	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*
1	-	$\mathcal{R}_{rnd}$	6.00	5.41	31.55	10.28	2.17	1.69	16.00	16.00	123.09	97.30	917.00	917.00
1	1	$\mathcal{R}_{rare}$	6.40	5.68	52.98	22.52	2.11	1.63	16.00	16.00	118.60	93.17	917.00	917.00
1	1	$\mathcal{R}_{freq}$	6.81	6.19	40.48	19.33	1.71	1.45	4.60	5.00	97.45	83.77	297.30	228.00
1	3	$\mathcal{R}_{rare}$	6.11	5.34	59.38	20.09	2.23	1.71	16.00	16.00	128.27	98.92	917.00	917.00
1	3	$\mathcal{R}_{freq}$	5.93	5.61	34.21	22.57	2.22	1.69	16.00	16.00	128.66	95.63	917.00	917.00
1	5	$\mathcal{R}_{rare}$	6.03	5.24	65.24	21.67	2.23	1.72	16.00	16.00	129.65	99.23	917.00	917.00
1	5	$\mathcal{R}_{freq}$	6.08	5.80	44.40	21.63	2.28	1.68	16.00	16.00	133.40	97.09	917.00	917.00
2	-	$\mathcal{R}_{rnd}$	6.86	6.72	26.46	13.36	1.70	1.41	5.00	5.00	94.71	79.12	292.20	228.00
2	1	$\mathcal{R}_{rare}$	7.18	6.42	54.07	19.02	1.64	1.41	5.00	5.00	92.57	81.27	270.90	228.00
2	1	$\mathcal{R}_{freq}$	7.43	6.71	29.58	20.77	1.58	1.36	4.40	2.10	87.86	78.98	307.30	217.70
2	3	$\mathcal{R}_{rare}$	7.56	7.14	60.67	21.65	1.55	1.34	5.00	5.00	87.66	76.78	271.50	228.00
2	3	$\mathcal{R}_{freq}$	8.26	7.46	36.17	20.00	1.46	1.31	3.70	2.20	81.30	73.98	229.00	197.50
2	5	$\mathcal{R}_{rare}$	6.99	6.43	49.71	23.15	1.74	1.44	6.60	5.00	98.38	83.35	382.20	228.00
2	5	$\mathcal{R}_{freq}$	7.15	6.50	28.82	22.01	1.72	1.44	5.00	5.00	97.29	83.02	319.00	228.00
3	-	$\mathcal{R}_{rnd}$	7.69	7.18	35.93	13.39	1.57	1.38	5.50	5.00	90.81	77.50	365.50	228.00
3	1	$\mathcal{R}_{rare}$	7.86	6.92	57.60	21.43	1.55	1.36	5.80	5.00	89.08	76.75	325.00	228.00
3	1	$\mathcal{R}_{freq}$	8.28	7.33	40.52	19.95	1.49	1.31	3.70	2.10	85.75	76.07	225.20	214.30
3	3	$\mathcal{R}_{rare}$	7.58	6.80	47.24	15.35	1.55	1.36	5.00	5.00	87.51	78.51	310.30	228.00
3	3	$\mathcal{R}_{freq}$	8.33	7.38	37.05	19.75	1.49	1.31	3.30	2.10	87.26	75.30	213.20	179.40
3	5	$\mathcal{R}_{rare}$	8.45	7.65	54.98	16.99	1.44	1.29	4.40	5.00	83.11	72.34	208.40	228.00
3	5	$\mathcal{R}_{freq}$	8.94	8.14	33.47	22.85	1.40	1.25	3.80	2.10	81.45	70.77	318.80	159.50

Table B.6: Measurements on $D_{Fish700}$ for both hard and soft constraints sessions

M	MB	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
			hc	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*
1	-	$\mathcal{R}_{rnd}$	15.77	11.88	118.76	22.25	1.74	1.37	17.00	17.00	355.90	277.94	1391.00	1391.00
1	1	$\mathcal{R}_{rare}$	17.23	12.13	162.98	35.73	1.65	1.33	17.00	17.00	333.52	263.68	1391.00	1391.00
1	1	$\mathcal{R}_{freq}$	15.67	13.57	69.35	40.90	1.43	1.20	5.00	7.00	291.58	241.17	1334.00	1355.00
1	3	$\mathcal{R}_{rare}$	16.94	11.34	183.32	31.80	1.76	1.37	17.00	17.00	358.54	274.30	1391.00	1391.00
1	3	$\mathcal{R}_{freq}$	15.16	12.45	77.08	38.65	1.75	1.38	17.00	17.00	356.86	275.48	1391.00	1391.00
1	5	$\mathcal{R}_{rare}$	16.45	11.22	203.34	33.30	1.75	1.38	17.00	17.00	353.52	276.01	1391.00	1391.00
1	5	$\mathcal{R}_{freq}$	14.28	12.74	55.75	39.63	1.74	1.37	17.00	17.00	357.94	275.81	1391.00	1391.00
2	-	$\mathcal{R}_{rnd}$	17.05	14.01	91.52	24.44	1.42	1.19	5.00	7.00	291.86	241.34	1107.00	1124.00
2	1	$\mathcal{R}_{rare}$	18.71	13.44	175.12	41.87	1.48	1.20	7.00	7.00	304.67	237.76	1124.00	1124.00
2	1	$\mathcal{R}_{freq}$	17.30	14.46	77.30	38.67	1.31	1.14	5.00	5.00	271.82	227.44	839.00	839.00
2	3	$\mathcal{R}_{rare}$	18.87	14.36	180.14	42.96	1.36	1.17	5.50	7.00	277.02	238.63	1107.00	1124.00
2	3	$\mathcal{R}_{freq}$	17.77	15.47	92.61	39.87	1.25	1.10	5.00	1.70	258.75	222.33	839.00	456.50
2	5	$\mathcal{R}_{rare}$	17.92	13.38	164.83	39.44	1.42	1.20	5.00	7.00	290.31	241.82	1107.00	1124.00
2	5	$\mathcal{R}_{freq}$	17.02	14.24	70.43	35.80	1.44	1.19	5.00	7.00	293.60	237.93	1107.00	1124.00
3	-	$\mathcal{R}_{rnd}$	18.06	14.67	82.83	22.53	1.37	1.15	5.00	5.00	275.62	235.68	1107.00	1107.00
3	1	$\mathcal{R}_{rare}$	18.58	14.42	171.55	44.50	1.35	1.17	5.00	7.00	284.12	234.95	1107.00	1124.00
3	1	$\mathcal{R}_{freq}$	18.05	15.24	77.65	42.13	1.26	1.11	5.00	1.70	255.51	222.36	839.00	442.70
3	3	$\mathcal{R}_{rare}$	18.67	14.26	160.48	36.25	1.37	1.17	5.00	7.00	283.24	233.55	1107.00	1124.00
3	3	$\mathcal{R}_{freq}$	17.84	15.05	88.95	40.97	1.26	1.11	5.00	1.70	255.76	218.84	839.00	454.70
3	5	$\mathcal{R}_{rare}$	20.14	15.18	196.24	38.99	1.30	1.13	5.00	5.00	268.75	229.08	1107.00	1107.00
3	5	$\mathcal{R}_{freq}$	18.93	16.06	90.42	38.56	1.20	1.08	3.00	1.50	240.78	218.27	533.00	454.40

Table B.7: Measurements on $D_{Fish10K}$ for both hard and soft constraints sessions

M	MB	Ranking Method	Mean Avg. Rank		Max Avg. Rank		Avg. Constr.		Max Constr.		Avg. Nav. Cost		Max Nav. Cost	
			hc	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*	*hc*	*sc*
1	-	$\mathcal{R}_{rnd}$	423.61	273.56	1778.88	667.60	2.37	1.85	6.00	16.00	2230.75	1764.60	5514.00	5599.00
1	1	$\mathcal{R}_{rare}$	470.35	267.50	3181.95	798.95	2.23	1.76	7.80	16.00	2115.65	1674.98	5514.00	5599.00
1	1	$\mathcal{R}_{freq}$	396.30	319.59	1989.80	1275.53	2.04	1.52	6.80	16.00	1897.59	1429.37	5514.00	5599.00
1	3	$\mathcal{R}_{rare}$	467.46	253.91	3310.18	801.63	2.39	1.87	7.80	16.00	2261.96	1793.83	5514.00	5599.00
1	3	$\mathcal{R}_{freq}$	387.67	287.33	3085.08	998.60	2.40	1.87	6.80	16.00	2226.87	1781.77	5514.00	5599.00
1	5	$\mathcal{R}_{rare}$	475.68	256.56	3114.75	801.90	2.43	1.87	7.60	16.00	2294.55	1769.26	5514.00	5599.00
1	5	$\mathcal{R}_{freq}$	379.76	288.83	2388.55	1229.27	2.40	1.87	6.00	16.00	2274.77	1765.03	5514.00	5599.00
2	-	$\mathcal{R}_{rnd}$	437.07	313.51	1661.10	668.03	1.98	1.51	5.00	5.00	1894.57	1427.92	5221.00	5221.00
2	1	$\mathcal{R}_{rare}$	468.63	299.03	3048.55	771.13	1.96	1.51	5.00	5.00	1900.23	1437.85	5511.00	5511.00
2	1	$\mathcal{R}_{freq}$	409.23	344.90	2111.80	1275.53	1.79	1.36	5.00	5.00	1675.10	1310.94	5221.00	5221.00
2	3	$\mathcal{R}_{rare}$	467.07	313.83	3106.10	819.20	1.87	1.43	5.00	5.00	1778.52	1388.19	5221.00	5221.00
2	3	$\mathcal{R}_{freq}$	425.37	357.66	2617.92	1327.50	1.65	1.33	4.80	5.00	1580.40	1268.55	4943.00	5221.00
2	5	$\mathcal{R}_{rare}$	454.66	294.61	3113.30	828.88	1.98	1.51	5.00	5.00	1886.31	1434.79	5221.00	5221.00
2	5	$\mathcal{R}_{freq}$	399.17	331.62	2150.28	1327.03	2.00	1.51	5.00	5.00	1901.30	1480.39	5221.00	5221.00
3	-	$\mathcal{R}_{rnd}$	429.09	327.73	1780.65	667.68	1.84	1.44	5.60	5.00	1754.21	1397.66	5221.00	5221.00
3	1	$\mathcal{R}_{rare}$	461.98	310.82	2934.95	896.87	1.85	1.44	5.00	5.00	1771.80	1420.93	5221.00	5221.00
3	1	$\mathcal{R}_{freq}$	422.66	353.24	2169.98	1229.70	1.68	1.33	4.40	5.00	1612.07	1268.91	4551.00	5221.00
3	3	$\mathcal{R}_{rare}$	466.89	315.59	3207.95	839.70	1.87	1.43	5.00	5.00	1769.56	1366.93	5221.00	5221.00
3	3	$\mathcal{R}_{freq}$	422.29	353.14	2118.67	1019.13	1.66	1.33	5.00	5.00	1593.81	1242.03	5221.00	5221.00
3	5	$\mathcal{R}_{rare}$	472.08	331.93	3217.00	837.80	1.74	1.38	4.80	4.00	1666.37	1325.32	5066.20	3834.00
3	5	$\mathcal{R}_{freq}$	424.67	365.11	2022.09	1025.93	1.60	1.28	4.20	4.00	1523.79	1232.47	4277.00	3831.00

Table B.8: Evaluation results in each dataset for $R = 10$ and $R = 20$

Dataset	HC_{size}	**Empty Answers (%)**		**Append-Blocks successses (%)**		**Avg. AppendBlocks Coverage (%)**	
		$R = 10$	$R = 20$	$R = 10$	$R = 20$	$R = 10$	$R = 20$
D_{Cars}	4	87.8	88.6	71.4	85.7	95.13	99.15
D_{Cars}	5	96.3	96	71.2	88.8	94.37	98.5
D_{Cars}	6	99	99	62.1	89.8	93.11	98.45
D_{Rest}	4	98.5	99	0.1	0.2	15.94	21.92
D_{Rest}	5	99.9	99.9	0.1	0.1	19.06	24.9
D_{Rest}	6	100	100	0	0	17.93	24.27
D_{Hotels}	4	95.9	96.2	11.5	17.6	62.57	69.78
D_{Hotels}	5	99	99.4	5.7	8.6	62.32	67.61
D_{Hotels}	6	99.9	99.9	2.5	3.8	61.39	66.87
$D_{Fish700}$	4	94.4	92.5	22.5	27.6	71.69	75.92
$D_{Fish700}$	5	97.9	98.4	11.2	18.2	68.25	73.41
$D_{Fish700}$	6	99.5	99.4	6.5	12.3	67.34	72.69
$D_{Fish10K}$	4	94.9	94.7	7.2	12.4	62.04	67.24
$D_{Fish10K}$	5	99.1	98.7	5	6.8	61.63	64.11
$D_{Fish10K}$	6	100	99.9	2.1	2.6	59.77	62.2