MixedSearch: An Interactive System of Searching for the Best Tuple with Mixed Attributes

Weicheng Wang[†] Min Xie[‡✉] Raymond Chi-Wing Wong[†]
Hong Kong University of Science and Technology[†] Shenzhen Institute of Computing Sciences[‡]
wwangby@connect.ust.hk xiemin@sics.ac.cn raywong@cse.ust.hk

***Abstract*—Identifying the best tuples in a large database for users has been a longstanding challenge in database community. Many interactive methods have been proposed to help users search for their best tuples in the database. Specifically, each user undergoes rounds of interaction. In each round, the user is presented with two tuples and is asked to pick the one s/he prefers more. Based on the user feedback, the user preference can be learned implicitly. Eventually, the best tuple w.r.t. the learned user preference is returned. Many systems have been designed for conducting interactive methods. However, they mainly restrict their settings on databases with numerical attributes, neglecting that in reality, databases can also be described by categorical attributes. Although there are some strategies to convert categorical attributes to numerical attributes, the conversion not only incurs poor efficiency, but also requires heavy interactive effort. In light of this, we developed an interactive system, called MixedSearch, and demonstrated that the system could find the best tuples for users in the database described by mixed attributes.**

***Index Terms*—mixed attributes; user interaction; data analytics**

I. Introduction

Nowadays, a database typically contains millions of tuples. Each tuple is described by multiple attributes, including numerical attributes and/or categorical attributes. The numerical attributes come with fixed orders on their attribute values, e.g., attribute "Price" (a lower price is better). In contrast, the categorical attributes lack fixed orders. Different users can have diverse preferences on attribute values. For instance, attribute "Transmission type" (some people might favor automatic cars, while others may prefer manual cars). To assist users in finding tuples that align their preferences, various *interactive methods* have been proposed. These methods can be applied in many domains, e.g., car purchase, house buying, and trip planning.

Take the car purchase scenario for illustration. Suppose that a user Alice wants to buy a car. Following the existing studies [1]–[5], Alice's preference could be represented by a *utility function*. This function captures Alice's trade-off among different attributes and attribute values. For example, Alice may be willing to pay more to buy an SUV rather than a sports car. Based on the utility function, each tuple is associated with a *utility* (i.e., a function value), and the tuple with the highest utility is regarded as the best tuple w.r.t. Alice's preference.

Since the users' trade-offs are usually implicit in the users' minds and cannot be explicitly provided, the main idea of the interactive methods is to learn the users' utility functions by interacting with users. The interaction involves multiple interactive rounds. In each round, users are presented with two tuples and are asked to indicate which one they prefer more. By accumulating the users' feedback, the interactive methods can implicitly learn the users' utility functions. When there is enough information, they suggest the tuples from the database that have the highest utilities w.r.t. the learned utility functions.

Based on these interactive methods, many existing systems [2], [6], [7] were designed. They could interact with users for a few rounds to identify their best tuples. However, these existing systems have limitations in use, since they mainly restrict their settings on numerical attributes. When a tuple is also described by categorical attributes, they have to convert each categorical value to a new numerical attribute by one-hot encoding. For example, if the cardinality (i.e., the number of possible values) of a categorical attribute "Country" is 100, there will be 100 additional numerical attributes created, which may incur poor efficiency and a huge number of interactive rounds.

Motivated by the need of managing both numerical and categorical attributes (i.e., mixed attributes), we study problem *Interactive Search with Mixed Attributes (ISM)*. The aim is to interactively find the user's best tuple from the database that is described by numerical and categorical attributes. We develop a novel system called MixedSearch, which is powered by our methods in [1] and has the following attractive features. Firstly, it efficiently supports a special case of ISM where each tuple is described by categorical attributes only. Secondly, it supports the general case of ISM where each tuple is described both numerical and categorical attributes. Finally, it provides user-friendly interfaces for users (1) to interact with the system, (2) to visualize how attributes affect each other, and (3) to check how tuples are pruned based on the learned preference.

In this paper, we demonstrate how to use MixedSearch to help users search for their best tuples in the database with mixed attributes. Our major contributions are summarized:

- We develop a system, MixedSearch, for solving ISM, i.e., interactive search on databases with mixed attributes.
- MixedSearch supports both the special and general cases of ISM, via efficient and effective search methods.
- The system provides several interfaces and supports novel functionalities, e.g, it shows how different attributes interact with each other based on the user feedback.
- We deploy the system on two databases (a used car and a NBA databases) with mixed attributes for demonstration.

In the following, we first introduce our system architecture in Section II and then demonstrate it in Section III. Finally, Section IV concludes this paper with possible future work.

II. System Architecture

The input to problem ISM is a set D of n tuples. Each tuple $p = (p_{cat}[1]; p_{cat}[2]; \ldots; p_{cat}[d_{cat}]; p_{num}[1]; \ldots; p_{num}[d_{num}])$ is described by d mixed attributes, including d_{cat} categorical and d_{num} numerical attributes ($d = d_{cat} + d_{num}$). Let $p_{num} = (p_{num}[1]; \ldots; p_{num}[d_{num}])$ and $p_{cat} = (p_{cat}[1]; \ldots; p_{cat}[d_{cat}])$.

Following [2], [6], [7], we model the user preference as a *linear utility function*, denoted by f. It is defined as follows.

$$f(p) = \sum_{i=1}^{d_{cat}} u_{cat}[i]h(p_{cat}[i]) + \sum_{j=1}^{d_{num}} u_{num}[j]p_{num}[j]$$

- Function $h : p_{cat}[i] \; ! \; \mathbb{R}_+$ maps each categorical value to a real number, indicating to what extent a user favors a categorical value, where a larger number is preferred.
- Element $u_{cat}[i]$ (resp. $u_{num}[j]$) measures the importance of the i-th categorical (resp. the j-th numerical) attribute to the user. For ease of representation, we denote the elements by two vectors: *the categorical utility vector* $u_{cat} = (u_{cat}[1]; \ldots; u_{cat}[d_{cat}])$ and *the numerical utility vector* $u_{num} = (u_{num}[1]; \ldots; u_{num}[d_{num}])$.
- Function value $f(p)$, called the *utility* of p w.r.t. f, represents how much a user favors tuple p. The tuple with the highest utility is regarded as the user's best tuple.

We assume *w.l.o.g.* the following: (a) $\sum_{j=1}^{d_{num}} u_{num}[j] = 1$ and call the domain of u_{num} *the numerical utility space* and (b) each numerical attribute is normalized to $(0; 1]$ and a larger value is more favored. Moreover, we use a function g_i to denote the product of $u_{cat}[i]$ and $h(p_{cat}[i])$. Specifically, $g_i : p_{cat}[i] \; ! \; \mathbb{R}_+$ is defined to be: $g_i(p_{cat}[i]) = u_{cat}[i]h(p_{cat}[i])$ for $i \; 2 \; [1; d_{cat}]$. In this way, we focus on learning $g_i(p_{cat}[i])$ as a whole, instead of $u_{cat}[i]$ and $h(p_{cat}[i])$ seperately.

A. Architecture Overview

Our interactive system, MixedSearch, is designed for the ISM problem. It employs two methods: *SP-Tree* and *GE-Graph*. The *SP-Tree* method works for the special case of ISM, and the *GE-Graph* method is proposed for the general case of ISM. Both methods follow the same interaction architecture [2], [6], [8]. Specifically, they interact with a user for rounds until they can find the user's best tuple. Each round consists of three components. (1) *Tuple selection*. Based on the user's previous answers, the methods select two tuples and ask the user to pick the one s/he prefers. For example, a user might be presented with two cars: (a) an SUV with a price \$2000 and (b) a sports car with a price \$5000. (2) *Information maintenance*. According to the user's choice, the methods update the information maintained for learning the user's preference. (3) *Stopping condition*. The methods check whether the stopping condition is satisfied. If so, they terminate the interaction process and return the result. Otherwise, they start a new interactive round. In the following, we show how methods *SP-Tree* and *GE-Graph* address each component discussed above. For lack of space, more technical details are available in [1].

B. ISM with Categorical Attributes Only

The *SP-Tree* method works on the databases described by categorical attributes only. It maintains tuples in a tree structure, called *categorical value tree*, or *C-Tree* in short. During each interactive round, two tuples are selected from the C-Tree and presented to a user (tuple selection). Based on the user's answer, the method updates the C-Tree by pruning the tuples that cannot be the best tuple (information maintenance). When there is only one tuple left in the C-Tree, the method returns that tuple as the answer (stopping condition).

The C-Tree has $d_{cat} + 2$ levels and satisfies the following properties. (1) The root is in the 0-th level. (2) Each node in the i-th level stores a categorical value in the i-th categorical attribute, where $i \; 2 \; [1; d_{cat}]$. (3) Each tuple in the database is recorded in a leaf (i.e., the node in the $(d_{cat} + 1)$-th level). There is only one simple path from the root to the leaf, and the categorical values of the tuple are stored in the path in order. Figure 1 shows an example of the C-Tree that is built upon tuples $p_1; p_2; p_3; p_4$ in Table I, where $d_{cat} = 2$.

Suppose that a user prefers p_1 to p_3. Intuitively, this means that the user prefers value B_1 than value B_2 in the second attribute, since p_1 and p_3 have the same value in the first attribute. Given this information, p_3 cannot be the best tuple, and thus, can be pruned from the C-Tree. Moreover, other tuples that cannot be the best tuple can be identified by some *derivation rules* and also pruned from the C-Tree. To exemplify, p_2 and p_4 have the same categorical value A_2 in the first attribute, while in the second attribute, the values of p_2 and p_4 are B_1 and B_2, respectively. Since value B_1 is preferred by the user, p_4 cannot be the best tuple and thus, it is also pruned from the C-Tree. The updated C-Tree is shown in Figure 2.

C. ISM with Mixed Attributes

The *GE-Graph* method considers the general case of ISM in which tuples are described by categorical and numerical attributes. It maintains the following data structures.

(1) A numerical utility range $R \quad \mathbb{R}^{d_{num}}$, that maintains the learned user preference on numerical attributes. Recall that $\sum_{i=1}^{d_{num}} u_{num} = 1$. The user's numerical utility vector u_{num} can be seen as a point in space $\mathbb{R}^{d_{num}}$. We maintain a polyhedron R in $\mathbb{R}^{d_{num}}$, called *numerical utility range*, which contains u_{num}. Initially, R is the entire numerical utility space, i.e., $R = fr \; 2 \; \mathbb{R}_+^{d_{num}} j \sum_{i=1}^{d_{num}} r[i] = 1g$. During the interaction, R will be gradually shrunk based on the user feedback.

(2) A relational graph G, which maintains the learned user preference on categorical attributes. For each tuple pair $p; q \; 2 \; D$ that have different values in at least one categorical attribute, we build a node v in G, containing (a) the set CV_1 of categorical values in p but not in q, (b) the set CV_2 of categorical values in q but not in p, (c) several *upper bounds* (*ub*) and *lower bounds* (*lb*), where each upper (resp. lower) bound quantifies the maximum (resp. minimum) utility difference between the categorical values of p and q, i.e., it describes to what extent the users prefer CV_1 to CV_2. For example, given a pair of tuples p_1 and p_2 in Table I, we build a node v_1 in Figure 3, where $CV_1 = fA_1g$ and $CV_2 = fA_2g$. The node v_1 also stores the upper/lower bounds (i.e., lb and ub) on the utility difference between the categorical values of p_1 and p_2, i.e.,

TABLE I: Database

p	$p_{cat}[1]$	$p_{cat}[2]$
p_1	A_1	B_1
p_2	A_2	B_1
p_3	A_1	B_2
p_4	A_2	B_2

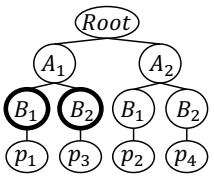

Fig. 1: C-Tree

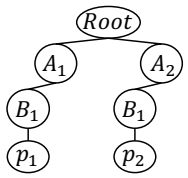

Fig. 2: Pruned C-Tree

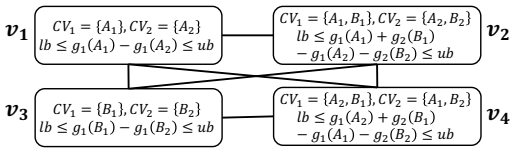

Fig. 3: Relational Graph

$g_1(A_1)$ $g_1(A_2)$. Note that multiple tuple pairs may share the same node in G if their CV_1 and CV_2 are the same. Moreover, each upper/lower bound is a scalar value computed based on the numerical utility vector u_{num}. Since u_{num} is uncertain in R, multiple upper bounds and lower bounds might be maintained in each node in G. Two nodes in G may be connected by an edge if the bounds they maintained can be used to derive more bounds on other nodes based on some *derivation rules*.

Here, we use two distinct data structures to handle numerical and categorical attributes, respectively, since these two types of attributes possess different characteristics (fixed orders vs. varied orders). Moreover, we maintain a tuple set C comprising tuples in D which are the candidates of the final answer. In each round, we select two tuples p and q from C (tuple selection). According to the feedback, we update R and G by considering two cases (information maintenance):

(a) If p and q are the same in all categorical attributes, we can construct a *hyper-plane* [1] based on the value difference among the numerical attributes of p and q. This hyper-plane will be used to shrink R to a smaller range.

(b) If p and q differ in at least one categorical attribute, we update the node v *2* G built for tuple pair p and q with new bounds based on the user preference between p and q. It may further trigger the update of other nodes in G.

Note that the updates on R and G interact with each other, i.e., the update on one may lead to the update on the other. Based on the updated R and G, we adopt several *pruning strategies* to prune from C the tuples that cannot be the best tuple. When there is only one tuple left in C, we stop the interaction and return the finally left tuple (stopping condition).

III. System Demonstration

We develop an interactive system called MixedSerach, which is built upon the techniques proposed. In the following, we demonstrate its functionality using a used car database as an example (our system also supports other databases, e.g., a NBA database) [1]. Each car in the database is described by seven attributes. The first three are categorical attributes (type, power, and transmission) with no fixed orders on their values. The last four are numerical attributes (price, year of manufacture, horsepower, used kilometers). In general, a lower price and fewer used kilometers are preferable, while a more recent manufacture date and a higher horsepower are preferable.

Our system, MixedSerach, can assist users in finding the best car in the database by interacting with them. Specifically, there are three interfaces as shown in Figure 4. (1) *Setup Interface*. It allows users to initialize the system setting (e.g., which car attributes should be considered). (2) *Categorical Attributes Only Interface*. It interacts with users if they only care about categorical attributes. (3) *Mixed Attributes Interface*. It interacts with users if they care about categorical and numerical attributes. Interested readers can find our source code and online system in [9], and demonstration video in [10].

A. Setup Interface

Figure 4(a) displays the setup interface. Similar to existing systems [2], [6], [7], it serves as the starting point of MixedSearch. The interface comprises four steps. Firstly, users are prompted to choose an interactive method. Method *SP-Tree* supports categorical attributes only, while method *GE-Graph* accommodates both numerical and categorical attributes. Note that if a user intends to consider numerical attributes alone, they can opt for any of the existing systems, which are beyond our focus. In the second step, users select the categorical attributes to be considered. The third step allows users to specify the numerical attributes to be considered, together with the acceptable value range on each attribute. For example, a user may pick attribute "Price" and set its acceptable range from 1000 USD to 50000 USD. Finally, in the fourth step, users define the maximum number of candidate cars to be selected. If users do not have any requirements regarding attributes or the number of cars, they can stick to the default setting and click the "Next" button to proceed.

B. Categorical Attributes Only Interface

This interface employs method *SP-Tree*, which specifically caters to categorical attributes, to interact with users. It is structured into three parts. To illustrate, consider a scenario where a user expresses his/her interest in two specific categorical attributes, namely power and transmission. Figure 4(b) depicts the interface for this particular use case.

(1) *Interaction.* In this part, MixedSearch interacts with users for rounds. In each round, it displays two cars to users and asks them to pick the one they favor more. For example, Figure 4 (b) shows the case after a user is asked one question, which shows the user two cars: Car 1 (Power: Diesel, Transmission: Auto) and Car 2 (Power: Diesel, Transmission: Manual). The user may click the "Choose" button on Car 2, indicating that s/he prefers Car 2 to Car 1. Based on the user feedback, another two cars are shown and the user is asked to provide an answer again, until the stopping condition is satisfied.

(2) *Visualization.* We plot the C-Tree to visualize the process of SP-Tree. Initially, each car in the database is recorded in a leaf. Every time a user answers a question, i.e., specifies his/her preference between a pair of cars, some branches of the C-Tree are pruned and thus, the number of leaves is reduced.