A Compositional Framework for Mining Longest Ranges

Haiyan Zhao[1], Zhenjiang Hu[1 2], Masato Takeichi[1]

[1] Department of Information Engineering
School of Engineering
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
113-8656 Tokyo, Japan
{zhhy,hu,takeichi}@ipl.t.u-tokyo.ac.jp
[2] PRESTO 21, Japan Science and Technology Corporation

Abstract. This paper proposes a *compositional* framework for discovering interesting range information from huge databases, where a domain specific query language is provided to specify the range of interest, and a *general* algorithm is given to mine the range specified in this language efficiently. A wide class of longest range problems, including the intensively studied optimized support range problem [FMMT96], can be solved systematically in this framework. Experiments with real world databases show that our framework is efficient not only in theory but also in practice.

1 Introduction

In this paper, we examine a new mining problem, called *longest range problem*, for discovering interesting range information from a huge database. It has promising applications in many trades, such as telecom, financial and retailing sectors. As an example, consider a relation [1] , namely *callsDetail*, in a telecom service provider database containing detailed call information. The attributes of the relation include *date*, *time*, *src_city*, *src_country*, *dst_city*, *dst_country* and *duration*, capturing concerned information about each call.

Some useful knowledge hidden in this relation can be used by the telecom service provider to make some decision. Suppose the telecom service provider is interested in offering a discount to customers. In this case, the timing of discount may be critical for its success, for example, to encourage the customers to make long time calls, it would be advantageous to offer it in a time interval in which the average calling time is not so long, say, less than 6 minutes. And the telecom service provider would want to do this campaign in a time interval as long as

[1] A relational database is a set of relations, while a relation (also called table) can be regarded as a set of tuples (or rows/records), and each tuple consists of several attributes (or fields).

possible, that is, to find a longest time interval that satisfies the above conditions. This can be described as

$$\begin{array}{l}\textbf{find longest } \mathit{time} \textbf{ range}\\ \textbf{from } \mathit{callsDetail}\\ s.t. \mathit{average}(\mathit{duration}) \leq 6.\end{array}$$

This kind of problems is practically important, but finding an efficient and correct algorithm is not easy. This has been seen by [FMMT96], which took great pains in solving *optimized range problem*, a special case of our longest range problems (see Section 4). Things turn out to be more interesting and difficult if we want to find the longest time interval under a more complicated condition. For the above example, to make the campaign more profitable, besides the average talking time is not long, the total time of talking during the time interval should be big enough, say, during one year the sum of *duration* is more than 10000 hours.

$$\begin{array}{l}\textbf{find longest } \mathit{time} \textbf{ range}\\ \textbf{from } \mathit{callsDetail}\\ s.t.\ \mathit{average}(\mathit{duration}) \leq 6\ \wedge\ \mathit{sum}(\mathit{duration}) \geq 10000 * 60\end{array}$$

However, as far as we know, this kind of problems has not been systematically solved yet, and it remains open how efficiently it can be solved.

This paper is therefore to address this kind of problems by making use of our work achieved in program calculation. In fact, the longest range problem is not a new problem at all; if we regard a relation in database as a list of records (tuples), it then boils down to the longest segment problem [Zan92,Jeu93] satisfying a given condition p, which is known in program calculation community.

Unfortunately, there does not always exist an efficient algorithm for computing the longest segment for any predicate p [Zan92]. But, if p has a particular shape or some nice properties, then a strong optimization can be made, resulting in an efficient algorithm using less than $\mathrm{O}(n^2)$ time or even linear time.

We hence propose a *compositional* framework to mine longest ranges by providing a class of predicates with nice property on which efficient solutions can be derived. The main contributions of this paper are as follows.

- We design a general querying language for mining the longest ranges. It is powerful and easy to learn, using a syntax similar to SQL. A wide class of longest range problems, including the optimized support range problem as intensively studied in [FMMT96,SBS99], can be expressed and solved by our language.
- We show that the language can be efficiently implemented by using techniques in program calculation. And moreover, our solution demonstrates that efficient algorithms for solving the longest range problems can be *compositionally* built according to the structure of predicates for specifying range properties. This composite approach is in sharp contrast with the existing case-by-case study as in [Zan92,Jeu93].

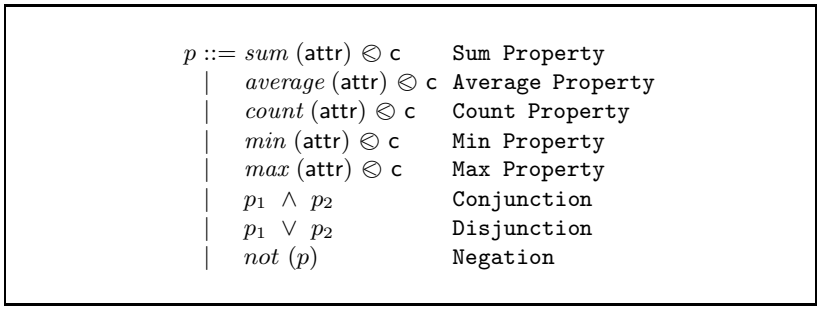

Fig. 1. Predicates for Specifying Range Property

– We apply this system to mining various range information with a POS database from a coffee shop with two years sales data. The experiments demonstrate that our framework is efficient not only in theory but also in practice, and can be applied in data mining field.

2 The Range Querying Language

The main statement **find** in our range query language resembles SELECT statement in SQL syntactically. It takes the form of

find longest *attr* **range**
from *tab*
where *property*

in which, **find**, **longest**, **range**, **from** and **where** are reserved words. Roughly speaking, it finds the longest range for the attribute *attr* from the relation *tab* under the condition given by *property*. Note that *attr* must be a numeric attribute of the relation *tab*.

It is evident that both *attr* and *tab* come from the database under consideration. However, how to specify *property* is not so easy as it looks. Just as discussed in Introduction, it cannot be guaranteed that there always exists an efficient algorithm to compute the longest range with respect to any *property*. It is thus crucial to define a language to describe proper *property*.

Predicates for Specifying Range Property. Since the property of range attribute is often described by aggregate functions of *sum*, *average*, *count*, *min* and *max* in data mining, we therefore design a class of predicates for defining the range properties of interest, as shown in Figure 1. Among which, $\trianglelefteq$ denotes a *transitive total order* relation, like $\leq$ or $>$. Our predicates are classified into two groups:

– five simple aggregate predicates specifying constraints on aggregate result on an attribute (field), including the range attribute, and

– three composite predicates combining predicates logically.

The former is used to specify simple properties, while the latter is for expressing more complicated properties in a compositional manner.

The general form of the simple predicates is

$$agg\ (\mathsf{attr}) \circledcirc \mathsf{c}$$

where agg is one of the aggregate functions, $\circledcirc$ a total transitive order, attr an attribute, and c a constant value. Instantiating $\circledcirc$ to be $\leq$, this property means that aggregate computation over the attribute attr in the range should be no more than constant c.

Example 1. In the telecom relation *callsDetail* given in Introduction, consider to find a longest time interval whose total calling time is less than 300 hours. We can specify this query just by

find longest *time* **range**
from *callsDetail*
where $sum\ (duration)\ \leq\ 300 * 60$.

The practical usage demands more interesting range properties rather than the simple aggregate ones. For this purpose, we provide composite predicates $\wedge$, $\vee$, and *not* in our language to combine predicates easily. As their names suggest, *not* denotes the logical negation of predicate, $\wedge$ is used to describe the logical conjunction of predicates, while $\vee$ is for the logical disjunction of predicates. The precedence of them is descending from *not* to $\vee$.

Example 2. Recall the example in Introduction, which can be specified in our language by

find longest *time* **range**
from *callsDetail*
where $average(duration) \leq 6\ \wedge\ sum(duration) \geq 10000 * 60$.

Consequently, our range query language, for its compositional feature, is powerful and easy for user to specify a wide class of longest ranges.

Remark. It is worth noting that we discuss only the properties related with computing the range, and omit the general selection conditions, which, in fact, can be easily filtered by preprocessing. For example,

find longest *time* **range**
from *callsDetail*
where $sum\,(\,duration) \leq 18000\ \wedge$
$src\_city = Tokyo\ \wedge\ dst\_city = Lübeck$

can be transformed to

find longest *time* **range**
from *callsDetail'*
where $sum\,(\,duration) \leq 18000$,

and *callsDetail'* is a view defined in SQL as

SELECT $*$
FROM *callsDetail*
WHERE *src_city* = *Tokyo* AND *dst_city* = *Lübeck*

3 Implementing the Range Querying Language

This section outlines how to implement our querying language efficiently. Our result can be summarized in the following theorem.

Theorem 1. The longest range specified in our language can be implemented using at most

$$\mathrm{O}(n \log^{k-1} n)$$

time, if every f and g used in the definition of primitive predicates inside property p can be computed in constant time. Here n denotes the number of tuples in the *tab* and k is a constant depending on the definition of *property* (Lemma 2). □

We prove this theorem by giving a concrete implementation, which consists of four phases: (i) bucketing the relation if necessary, (ii) normalizing the range property, (iii) refining the longest range problem, (iv) computing the longest range.

The detail of the implementation can be found in [ZHM02a]. For the reason of space, we only illustrate how to normalize the range property and what the core of our problem is after normalization.

To normalize the range property specified by user, we first give the following definition.

Definition 1 (Primitive Predicate). A primitive predicate takes the form of

$$\mathsf{f}\ (\mathit{head}) \circledS \mathsf{g}\ (\mathit{last}),$$

where *head* and *last* indicate the first and the last element of a given range respectively, f and g be any functions applied to the end elements. □

Semantically, it means that the leftmost element (after applied function f) of a range has a *transitive total order* relation $\circledS$ with the rightmost element (after applied function g).

Lemma 1. All the simple predicates given in Figure 1 can be represented in the form of primitive predicate. □

We omit the proof for this lemma here, and just demonstrate it by an example. Recall the condition in example 1: $sum\,(\,xs) \leq 18000$. How to eliminate this *sum* function? The trick is to do preprocessing like

$$s_i \;=\; s_{i-1} \;+\; x_i$$

to compute every prefix sum of the input list xs and get a new list ss:

$$\begin{array}{ll} xs: & [x_1, x_2, \ldots, \underline{x_h}, \ldots, \underline{x_l}, \ldots] \\ ss: & [s_1, s_2, \ldots, \underline{s_h}, \ldots, \underline{s_l}, \ldots]. \end{array}$$

Therefore to compute the sum of a range $xs' = [x_h, \ldots, x_l]$, we can now use the end elements of xs' and corresponding ones of ss', i.e., x_h, s_h, x_l, s_l:

$$sum\ (xs') = x_h + (s_l - s_h).$$

Thus, $sum\ (xs') \leq 18000$ is coded as $x_h - s_h \leq 18000 - s_l$, a relation between the two end elements of the preprocessed segment. Accordingly, for any list xs, we can do this preprocessing to get a new list with each element x_i changed to a pair (x_i, s_i), and reduce aggregate *sum* property to a transitive total order relation between the leftmost and rightmost elements of the concerned segment. It is worth noting that this preprocessing does not raise additional cost by using accumulation and fusion technique [Bir84].

With the same trick, the other four aggregate predicates can also be normalized into the primitive form. Accordingly, we further have the following self-evident lemma hold for the composite property.

Lemma 2 (Disjunctive Normal Form). Any composite predicate can be expressed in its canonical form, that is,

$$\begin{array}{rcll} p\ (x) & = & p_{11}\ (x) \wedge p_{12}\ (x) \wedge \ldots \wedge p_{1k_1}\ (x) & \vee \\ & & p_{21}\ (x) \wedge p_{22}\ (x) \wedge \ldots \wedge p_{2k_2}\ (x) & \vee \\ & & \ldots\ldots & \vee \\ & & p_{m1}\ (x) \wedge p_{m2}\ (x) \wedge \ldots \wedge p_{mk_m}\ (x) & \end{array}$$

where, p_{ij} is primitive predicate, and the maximum of k_1, k_2, ..., k_m is exactly the k in Theorem 1. □

Lemma 2 shows that a range property specified by our language can be normalized into its canonical form, that is, a disjunction of simpler components, which is either a primitive predicate or a conjunction of several primitive ones. If we can address both the primitive and the conjunction case, The result for the disjunction case can be gotten easily by calculating that for each component and selecting the longest one as the result. Thus, the crucial part is how to deal with conjunction case with primitive case as its special.

Semantically, the conjunction case is to compute the longest range that satisfies a number of primitive predicates simultaneously, i.e.,

$$\begin{array}{rll} p\ z = & f_1\ (\mathit{head}) \circledcirc g_1\ (\mathit{last}) & \wedge \\ & f_2\ (\mathit{head}) \circledcirc g_2\ (\mathit{last}) & \wedge \\ & \ldots\ldots & \wedge \\ & f_k\ (\mathit{head}) \circledcirc g_k\ (\mathit{last}) & \end{array}$$

where k is the number of primitive predicates. If we capsule this composite conjunction as below by tupling all of its primitive component together,

$$(f_1, \ldots, f_k)\ (\mathcal{R}_1, \ldots,\ \mathcal{R}_k)\ (g_1, \ldots, g_k)$$

with each $\mathcal{R}_i$ is a *transitive total order* relation, and

$$(x_1, ..., x_k)\ \ (\mathcal{R}_1, ..., \mathcal{R}_k)\ (y_1, ..., y_k) \equiv \wedge_{i=1}^{k} x_i\ \mathcal{R}_i\ y_i$$

then what we need to implement boils down to the following problem:

> Given a list, compute the length of a longest nonempty range such that the computation on the leftmost element is related with that on the rightmost element by a relation $\bar{\mathcal{R}}$ (which is not necessary to be a total order), that is,
>
> $$\bar{f}\ (\mathit{head})\ \bar{\mathcal{R}}\ \bar{g}\ (\mathit{last})$$
>
> where $\bar{f} = (f_1, \cdots, f_k)$
> $\bar{g} = (g_1, \cdots, g_k)$.

Fortunately, this refined problem can be solved in $\mathrm{O}(n \log^{(k-1)} N)$ time by using our algorithm given in [ZHM02b,ZHM02a]. Theorem 1 has accordingly been proved. Especially, for the case of $k = 1$ (primitive case), the longest range can be computed in linear time.

4 An Application: Optimized Support Range Problem

We present an application of our framework for practical data mining. As a special case of our longest range problem, the optimized support range problem, first studied in [FMMT96], is very useful in extracting correlated information. For example, the optimized association rule [2] for *callsDetail* in the telecom database,

$$(\mathit{date} \in [d_1..d_2]) \wedge (\mathit{src\_city} = \mathit{Tokyo}) \ \Rightarrow\ (\mathit{dst\_city} = \mathit{Lübeck})$$

describes the calls from *Tokyo* during the date $[d_1, d_2]$ are made to *Lübeck*. Suppose that the telecom service provider wants to offer discounts to *Tokyo* customers who make calls to *Lübeck* at a period of consecutive days in which the maximum number of calls from *Tokyo* are made and a certain minimum percentage of the call from *Tokyo* are to *Lübeck*. This is known as the optimized support range problem, which maximizes the *support* of the given optimized association rule with the *confidence* of the rule exceeds a given constant θ.

To do this, first we preprocess the original relation by adding a new attribute called *support*. It is defined according to the above rule by

$$\mathit{support} = \begin{cases} 1 & \mathit{src\_city} = \mathit{Tokyo} \ \wedge\ \mathit{dst\_city} = \mathit{Lübeck} \\ 0 & \text{otherwise} \end{cases}$$

[2] The definition for optimized associate rule and its associate properties *support* and *confidence* can be found in [FMMT96].

After bucketing *callsDetail* according to the range attribute *date*, it is just to compute the longest *date* range with respect to the average of *support* no less than the given θ. Thus, the optimized support range problem is only a special case of the longest range problem, it can be simply expressed by our language as follows.

find longest *date* **range**
from *callsDetail*
where *average* (*support*) $\geq \theta$

From Theorem 1, we know that it can be solved in $O(n)$ time.

5 Conclusion

In this paper, we identify one important class of data mining problems called longest range problems, and propose a compositional framework for solving the problems efficiently. This work is a continuation of our effort to investigate how program calculation approach could be used in data mining [HCT00], and it confirms us with its promising result.

As a future work, we want to investigate how to extend our framework to mining multiple range attributes efficiently.

For detailed explanation and experimental results, please refer to [ZHM02a].

References

[Bir84] R. Bird. The promotion and accumulation strategies in transformational programming. *ACM Transactions on Programming Languages and Systems*, 6(4):487–504, 1984.

[FMMT96] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. In *Proc. ACM PODS'96*, pages 182–191, Montreal Quebec, Canada, 1996.

[HCT00] Z. Hu, W.N. Chin, and M. Takeichi. Calculating a new data mining algorithm for market basket analysis. In *Proc. of PADL2000, LNCS 1753*, pages 169–184, Boston, Massachusetts, January 2000. Springer-Verlag.

[Jeu93] J. Jeuring. *Theories for Algorithm Calculation*. Ph.D thesis, Faculty of Science, Utrecht University, 1993.

[SBS99] R. Rastogi S. Brin and K. Shim. Mining optimized gain rules for numeric attributes. In *Proc. of ACM KDD'99*, 1999.

[Zan92] H. Zantema. Longest segment problems. *Science of Computer Programming*, 18:36–66, 1992.

[ZHM02a] H. Zhao, Z. Hu, and M.Takeichi. A compositional framework for mining longest ranges. Technical Report METR 02-05, Department of Mathematcical Engineering, Univ. of Tokyo, May 2002. Document is available at ftp://www.ipl.t.u-tokyo.ac.jp/~zhhy/pub/metr0205.ps.

[ZHM02b] H. Zhao, Z. Hu, and M.Takeichi. Multidimensional searching trees with minimum attribute. *JSSST Computer Software*, 19(1):22–28, Jan. 2002.