Binary Time-Series Query Framework for Efficient Quantitative Trait Association Study

Hongfei Wang and Xiang Zhang
Department of Electrical Engineering and Computer Science
Case Western Reserve University
Cleveland, Ohio 44106
{hongfei.wang, xiang.zhang}@case.edu

***Abstract*—Quantitative trait association study examines the association between quantitative traits and genetic variants. As a promising tool, it has been widely applied to dissect the genetic basis of complex diseases. However, such study usually involves testing trillions of variant-trait pairs and demands intensive computational resources. Recently, several algorithms have been developed to improve its efficiency. In this paper, we propose a framework, Fabrique, which models quantitative trait association study as *querying binary time-series* and bridges the two seemly different problems. Specifically, in the proposed framework, genetic variants are treated as a database consisting of binary time-series. Finding trait-associated variants is equivalent to finding the nearest neighbors of the trait. For efficient query process, Fabrique partitions and normalizes the binary time-series, and estimates a tight upper bound for each group of time-series to prune the search space. Extensive experimental results demonstrate that Fabrique only needs to search a very small portion of the database to locate the target variants and significantly outperforms the state-of-the-art method. We also show that Fabrique can be applied to other binary time-series query problem in addition to the genetic association study.**

Keywords*—*Quantitative Trait Association Study, Time-Series Query, Upper bound, Lower bound, Pruning.

I. Introduction

A fundamental problem in life science is the characterization of genetic factors that underlie the differences in complex traits. An enormous amount of genetic variants have been identified and cataloged through various genome sequencing projects [1]. The most abundant source of genetic variation is represented by single nucleotide polymorphism (SNP), which is a DNA sequence variation occurring when a single nucleotide (A, T, G, or C) in the genome differs between individuals of a species. SNPs can usually be represented as binary variables in haploid organism (with one set of chromosomes). Rare variants in diploid cells can also be considered to be binary [2].

Quantitative trait loci (QTL) study analyzes SNPs across a population in order to find SNPs that are significantly associated with the quantitative traits, such as blood pressure or tumor size. Recently, genome-wide study of expression quantitative trait loci (eQTL) has emerged as a prominent tool to dissect genetic basis of gene expression [3]–[5]. In an eQTL study, gene expression profiles are treated as traits. Its goal is to find significant associations between the SNPs and the gene expression traits. The outcome of these studies can lead to better understanding of genetic effects, dissecting gene regulatory network, and ultimately uncovering underlying biochemistry [6].

In eQTL studies, the association between each expression trait and each SNP is assessed by a statistical test (e.g., F-test or t-test) [7]. The large number of SNP-trait tests leads to the multiple testing problem [8], which usually needs to be corrected by permutation test [9], [10]. This process imposes great computational burden. Consider a study with 100,000 SNPs and 10,000 traits, and assume 1,000 permutations are performed for statistical significance assessment. The total number of SNP-trait tests is a trillion (10^{12}). This number can be much larger with more SNPs and gene expression traits becoming available [1].

Recently, several methods have been developed to address the computational issues in quantitative trait association studies [11]–[14]. PLINK [11] is a standard tool for various genetic study tasks. However, it does not scale well to large datasets [14]. FastMap [13] uses a summation tree to organize the SNPs to achieve incremental computing of the trait-SNP association scores. SnpMatrix [12] and Matrix eQTL [14] utilize fast matrix operations to improve the computational efficiency. Despite their success, these methods are still very time-consuming for large-scale studies.

In this paper, we propose a novel framework, Fabrique (Fast binary time-series query), to address the computation challenge in quantitative trait association study. Fabrique formulates the problem as the following time-series query problem. The entire set of SNPs constitutes a database, each of which corresponds to a binary time-series. A quantitative trait is regarded as a query. Given a query, the goal is to identify SNPs that are the nearest neighbors of the query under certain distance metric. For efficient query, Fabrique first partitions SNPs into different groups based on Z-normalization. A tight upper bound on the distance between the SNPs and the query trait can be devised for each group of SNPs. Next, SNP groups are examined in descending order of their upper bounds. The search space can be pruned if the upper bound of the remaining groups is less than the nearest distance identified so far. The contributions of this paper are summarized as follows.

I. To the best of our knowledge, this is the first work to formulate quantitative trait association study from a time-series query perspective. This formulation bridges two seemly different problems and enables to leverage mature techniques developed for time-series query in the data mining community to address an important genetics problem.

II. By exploiting the characteristics of binary time-series, we devise tight bound to prune the search space. We provide theoretical analysis to show that the developed bound is tighter than the existing ones [15]–[17]. The devised bounds can be employed in other applications, such as template matching, in addition to the targeted genetic association study.

III. Experimental results show that the proposed Fabrique method only exams a small fraction of all the SNPs before identifying the nearest one and occupies a small amount of memory space. It achieves one order of magnitude speed up compared to the state-of-the-art method for quantitative trait association study.

The rest of the paper is organized as follows. Section II gives a brief survey of the related work. Section III presents the problem formation and other preliminaries. Section IV, V, and VI formally describes the proposed framework. Experimental results are shown in Section VII. Section VIII summarizes our work.

II. Related Work

Developing efficient algorithms for quantitative trait association study has recently attractive intensive research interests. Existing methods [11]–[14] mainly tackle the computational challenges from the perspective of statistical testing itself. The widely used PLINK [11] is a comprehensive tool set designed for different types of genetic association studies. Its computational efficiency is not optimized and usually requires clusters or very long running time for large scale studies. In FastMap [13], a summation tree connecting SNPs for incremental computing is proposed. However, many synthetic SNPs need to be added to the summation tree to ensure that all real SNPs can be connected by the tree. Other methods [12], [14] utilize efficient matrix operations encoded in Matlab or R to speedup the process. The optimization techniques of these methods are limited to the code/program level.

Time-series has been extensively studied in the data mining community [18]. In the pioneer work [19], time series has been studied in the frequency domain by applying Discrete Fourier Transform. Various dimension reduction, indexing and bounding techniques have been proposed since then [15], [20]. In [21], different bounding methods are examined and many of them are proven to be effective for mining time-series data. In this paper, we show that *binary* time-series has unique properties that can be unitized to design efficient algorithm for quantitative trait association study.

Querying binary time-series also has other applications. One example is template matching, which is a key problem in image processing [22], [23]. It identifies the part of an input image that best matches a given template. This task is computationally challenging because every subset of an input image needs to be compared with the template. Much effort has been made to reduce the search space by applying bounding techniques [17], [24]. Our method can also be applied to this problem setting. We formally prove that proposed bound is tighter than the ones that are specifically designed for binary template matching.

III. Problem Formulation

Suppose that there are a set of M binary SNPs $\{\mathrm{X}^j\}$ ($1 \leq j \leq M$) and a quantitative trait Y, and the number of individuals is N. We use X^j_i ($1 \leq i \leq N$) to represent the i-th value of X^j. The goal of quantitative trait association study is to find the SNPs that are most highly associated with Y. The SNP-trait association is usually determined by a statistical test such t-test, F-test, or Linear Regression (LR). It has been shown that all these tests are equivalent to correlation [14], which is defined as

$$Corr(\mathrm{X}^j, Y) = \frac{\frac{1}{N}\sum_{i=1}^{N} \mathrm{X}^j_i Y_i - \mu_{\mathrm{X}^j}\mu_Y}{\sigma_{\mathrm{X}^j}\sigma_Y}, \quad (1)$$

where μ_{X^j} and σ_{X^j} are the mean and standard deviation of X^j, respectively.

We can formulate quantitative trait association study as the following binary time-series query problem. Let $\{\mathrm{X}^j_i\}$ be a set of M time series, each consisting of N binary values ($1 \leq i \leq N$, $1 \leq j \leq M$). For a given query Y, the task is to find the nearest neighbor X^{j*} of Y using correlation as the similarity measure. An example dataset containing 8 SNPs over 7 individuals is shown in Table II.

With the above problem formulation, the proposed Fabrique framework consists of the following two major components.

I. A **preprocessing** component that performs partitioning and Z-normalization to the binary time-series.

II. An **estimation** component that computes the upper bound for each subset of time-series.

The next two sections will describe these two components in detail. Some frequently used notations are collectively provided in Table I.

IV. Preprocessing the binary time series

A. Applying Z-normalization

The proposed Fabrique framework applies Z-normalization to both the binary time series and the query. Normalization adjusts data values to a common range and can (partially) eliminates the impact of the data points with extreme values [21]. Applying Z-normalization also brings in the following advantages that are specific to quantitative trait association study.

I. Under Z-normalization, using correlation as the similarity measure is the same as using many other commonly use test statistics such as t-test, F-test, and Linear Regression [14].

II. After Z-normalization, the time-series and query both have mean 0 and standard deviation 1. Equation 1 can be simplified as

$$Corr(X^j, Y) = \sum_{i=1}^{N} X^j_i Y_i. \quad (2)$$

Comparing to Equation 1 without normalization, Equation 2 can be efficiently computed by leveraging matrix multiplication.

TABLE I. FREQUENTLY USED NOTATIONS.

Notations	**Interpretations**
$\{\mathrm{X}_i^j\}$	A set of M time series, whose lengths are N ($1 \le i \le N$, $1 \le j \le M$)
X_i^j	Z-normalized value of X_i^j, from the i^{th} datapoint and j^{th} time series
X_{N1}^j (X_{N0}^j)	The values of normalized 1 (0) for the j^{th} binary time series
C_1^j	Equals to $\sum_{i=1}^N \mathrm{X}_i^j$, the number of 1's in X^j
$EuD(X^j, Y)$	The Euclidean distance between a time series X^j and a query Y
$Corr(X^j, Y)$	The correlation between a time series X^j and a query Y
S_l	A set of binary time series with the same $C_1^j$for $\forall$ $\mathrm{X}^j \in S_l$
S_k	A set of binary time series with identical datapoints for the corresponding I_P
T_k	A tuple consists of S_k, and $X_{N1}^{S_k}, X_{N0}^{S_k}$
LB_Keogh	The lower bounding function introduced by [21]
UB_Fab	The upper bounding function proposed in this work
LB_Fab_{EU}	The variant of UB_Fab; the lower bounding function measured by Euclidean distance
UB_EBC	The upper bounding technique used in the EBC algorithm proposed in [17].

TABLE II. EXAMPLE OF APPLYING Z-NORMALIZATION AND PARTITIONING TO THE BINARY TIME SERIES

	A sample of binary time series								
j	**Original** $\{\mathrm{X}^j\}$	**Z-normalized** $\{X^j\}$							
1	1 **0** 0 1 **0** 0 1	1.15	**-0.87**	-0.87	1.15	**-0.87**	-0.87	1.15	
2	0 **1** 0 0 **0** 0 1	-0.63	**1.58**	-0.63	-0.63	**-0.63**	-0.63	1.58	☆
3	1 **0** 1 1 **0** 0 0	1.15	**-0.87**	1.15	1.15	**-0.87**	-0.87	-0.87	
4	0 **0** 0 1 **0** 1 1	-0.87	**-0.87**	-0.87	1.15	**-0.87**	1.15	1.15	
5	0 **1** 1 1 **0** 0 0	-0.87	**1.15**	1.15	1.15	**-0.87**	-0.87	-0.87	❊
6	1 **0** 0 0 **0** 1 1	1.15	**-0.87**	-0.87	-0.87	**-0.87**	1.15	1.15	
7	0 **1** 1 0 **0** 0 0	-0.63	**1.58**	1.58	-0.63	**-0.63**	-0.63	-0.63	☆
8	0 **0** 1 0 **0** 1 1	-0.87	**-0.87**	1.15	-0.87	**-0.87**	1.15	1.15	

Table II shows an example of Z-normalization. It contains eight binary time series. Each has seven data points ($N = 7$, $M = 8$). Let X^j (*italicized*) be the Z-normalized version of X^j. Let X_{N1}^j (X_{N0}^j) denote the normalized values of 1 (0) for the j^{th} binary time series. For instance, X_{N1}^1 = 1.15, X_{N0}^2 = -0.63. Marked by ☆, $\{X^2, X^7\}$ share the same $\{X_{N1}, X_{N0}\}$, but are different from the other six X^j's. Such distinction stems from the heterogeneous numbers of 1's in $\{\mathrm{X}^j\}$, as both X^2, X^7 have two 1's, whereas each of the $\{\mathrm{X}^1, \mathrm{X}^3, \mathrm{X}^4, \mathrm{X}^5, \mathrm{X}^6, \mathrm{X}^8\}$ has three.

1) Normalizing A Binary Time Series: Z-normalizing a single time series X^j consists of computing (i) the mean $\overline{\mathrm{X}}^j = \frac{1}{N}\sum_{i=1}^N \mathrm{X}_i^j$, (ii) the sum of the squared deviation $SS_{\mathrm{X}^j} = \sum_{i=1}^N (\mathrm{X}_i^j - \overline{\mathrm{X}}^j)^2$, and (iii) for each of the data-point value, $X_i^j = (\mathrm{X}_i^j - \overline{\mathrm{X}}^j)/\sqrt{\frac{SS_{\mathrm{X}j}}{N-1}}$, for $i = 1, 2, \cdots, N$.

On the contrary, binary data simplifies the steps as the following,

- In Step (ii), for a binary time series X^j of length N, computing the sum of squared deviation can be done by $SS_{\mathrm{X}^j} = (1 - \overline{\mathrm{X}}^j)^2 \times C_1^j + (0 - \overline{\mathrm{X}}^j)^2 \times (N - C_1^j)$, where $C_1^j$is the number of 1's in X^j (C_1^j= $\sum_{i=1}^N \mathrm{X}_i^j$). Therefore,
$$SS_{\mathrm{X}^j} = C_1^j \times (1 - C_1^j/N). \qquad (3)$$
Since C_1^j can be obtained through Step (i), there is no need to scan through N data points in X^j to compute SS_{X^j}. The complexity is $O(1)$.
- In Step (iii), two runs of centering operation are necessary and sufficient, i.e., converting 1 to X_{N1}^j and 0 to X_{N0}^j, respectively. The complexity is again $O(1)$ and does not depend on N.

2) Normalizing A Set of Binary Time Series: For generic time series with continuous values, the above three-step routine has to be performed for every time series in the database. Binary data dramatically alleviates the computation burden. Z-normalizing a set of time series with the same C_1^j (j is the index of a time series) produces only two distinct Z-normalized values, namely, X_{N1}^j and X_{N0}^j. Hence, Z-normalization only needs to be performed once for the set of time series with the same C_1^j .

Consider a set of time series, $\{\mathrm{X}^1, \mathrm{X}^3, \mathrm{X}^4, \mathrm{X}^5, \mathrm{X}^6, \mathrm{X}^8\}$, from Table II. We choose an arbitrary one for Z-normalization, say X^1. The resulting X_{N1}^1 and X_{N0}^1 satisfy
$$X_{N1}^1 = X_{N1}^{\{3:6,8\}}, \quad X_{N0}^1 = X_{N0}^{\{3:6,8\}}.$$
The equation holds since these six time series have the same C_1^j, which leads to a common $\overline{\mathrm{X}}^j$ and SS_{X^j} among them. To normalize the remaining time series $\{\mathrm{X}^3, \mathrm{X}^4, \mathrm{X}^5, \mathrm{X}^6, \mathrm{X}^8\}$, the task reduces to merely substituting the 1 (0) for X_{N1}^1 (X_{N0}^1). Consequently, the complexity for Z-normalizing all the binary time series (with several distinct C_1^j's) only depends on the number of different C_1^j's within, which is less than the length N and does not depend on M. The substitution procedure can be postponed to a very late stage and may eventually be saved. Further details will be discussed in Section VI.

B. Partitioning Data Set

We can partition the time series into a collection of sets $\{S_l\}_{l\in L}$ ($L = \{1, 2, \cdots\}$) according to the different C_1^j. Each subset is characterized by an unique pair of (X_{N1}^j, X_{N0}^j). An upper bound on the correlation between SNPs and trait can be established based on the (X_{N1}^j, X_{N0}^j) value. (Bound estimation will be discussed in Section V.) However, given that $N \ll M$, there may be many X^j's in the resulting subsets. To reduce the number of time-series in each subset and tighten the bound, the proposed Fabrique method partition the data as follows. It randomly selects P data point indices ($1 < P < N$) to further partition $\{S_l\}_{l\in L}$ into a total of K subsets ($1 < K < M$), such that times series with identical values for these P indices are assigned into the same subset. Let I_P and I_Q be the index sets containing the prefixed data points from a random selection, and the remaining data points, respectively. A subset S_k (assuming $|S_k| \geq 2$) satisfies that

$$\forall\ u \neq v, \mathrm{X}^u, \mathrm{X}^v \in S_k, \text{then } \mathrm{X}_p^u = \mathrm{X}_p^v, \text{for all } p \in I_P.$$

Table II also shows an example of applying our partition method to the set of binary time series. According to a randomly selected I_P ({2,5}), $\{X^j\}$ is partitioned into three subsets: the largest one is $\{X^1, X^3, X^4, X^6, X^8\}$; the other two are denoted by ☆ and ✻, respectively. Note that I_P is not directly applied on the un-normalized $\{\mathrm{X}^j\}$, otherwise the fifth time series (denoted by ✻) would not be set apart from the ones denoted by ☆.

This example reveals several properties related to the resulting subsets. First, binary time series with the different C_1^jnever reside in the same subset. Second, those with the same C_1^jmay not be assigned to the same subset either. Only when they have identical data point values for I_P, they will be partitioned into the same subset. Third, each subset is characterized by an unique pair of (X_{N1}^j, X_{N0}^j).

Now consider the scenario when time series become available in an incremental fashion. For a new time series, if its values for I_P match those of any subset already exists, it is then assigned to the matched one. Otherwise a new subset is created to contain this time series. The cost for updating is very limited.

C. Preprocessing_Fabrique

Algorithm 1 summarizes the Z-normalization and partition steps in the preprocessing procedure of Fabrique. It partitions the data in two steps. The first is based on the C_1^j, thus characterizing each set with an unique pair of (X_{N1}^j, X_{N0}^j). The second employs a prefixed data point set I_P to bring down the dimensionality from N to Q, where $Q = N - P$, and reduces the number of time series in the resulting subsets.

Specifically, in Algorithm 1, the C_1^j is counted for each of the M binary time series (Lines 1-3). Time series are then partitioned into L sets $\{S_l\}_{l\in L}$ so that within the same set, C_1^j's are identical (Line 4). Next, P data points are randomly selected (Line 5). The algorithm then navigates through $\{S_l\}_{l\in L}$ (Line 7); for one S_l, it performs Z-normalization on an arbitrary time series within (Line 8), further divides a $\{S_l\}$ into subsets with homogenous bits for the P bits selected (Line 9), and organizes the resulting subsets into a number of tuples (Lines 10 -14), whose sum is K as the results for Algorithm 1 (Line 16). Note that Z-normalization is performed only $2|L|$ times, according to Lines 7 and 8. The complexity is $O(N)$.

Algorithm 1: Preprocessing_Fabrique

```
Input: A set of time series
       $\{\mathrm{X}_i^j\}, 1 \leq i \leq N, 1 \leq j \leq M$
Output: $|K|$ tuples, as $T_k = \langle S_k, X_{N1}^{S_k}, X_{N0}^{S_k}\rangle, k \in K$
1  for $j \leftarrow 1$ to $M$ do
2      Obtain $C_1^j$ for $\mathrm{X}_1^j$ ;
3  end
4  Partition $\{\mathrm{X}_i^j\}$ into a collection of sets $\{S_l\}_{l\in L}$ indexed
   by an index set $L = \{1, 2, \cdots\}, |L| \leq N, s.t.$
   $(a)\ \forall\ u \neq v, \mathrm{X}^u, \mathrm{X}^v \in S_l, C_1^u = C_1^v,$ and $(b)\ \forall\ l \in L, l' \in L \setminus \{l\}, \forall\ \mathrm{X}^u \in S_l, \mathrm{X}^{u'} \in S_{l'}, C_1^u \neq C_1^{u'}$ ;
5  Randomly select a prefixed index set $I_P$. The remaining
   indices form another set
   $I_Q, s.t.\ I_Q = \{1, 2, \cdots, N\} \setminus I_P$;
6  $k \leftarrow 0$;
7  for $l \leftarrow 1$ to $|L|$ do
8      Compute $X_{N1}^{S_l}, X_{N0}^{S_l}$ using $\forall\ C_1^j, s.t.\ \mathrm{X}_1^j \in S_l$;
9      Partition $S_l$ into
       $\{S_a\}_{a\in A}, A = \{1, 2, \cdots\}, |A| \leq |L|,$
       $s.t.\ \forall\ u \neq v, \mathrm{X}^u, \mathrm{X}^v \in S_a, \mathrm{X}_p^u = \mathrm{X}_p^v,$ for all $p \in I_P$;
10     for $a \leftarrow 1$ to $|A|$ do
11         $k \leftarrow k + 1, S_k \leftarrow S_a$;
12         $X_{N1}^{S_k} \leftarrow X_{N1}^{S_l}, X_{N0}^{S_k} \leftarrow X_{N0}^{S_l}$;
13         $T_k \leftarrow \langle S_k, X_{N1}^{S_k}, X_{N0}^{S_k}\rangle$;
14     end
15 end
16 return a collection of tuples $\{T_k\}_{k\in K}, |K| \leq M, I_P, I_Q$
```

The tuple structure T_k captures all the necessary information for a subset of binary time series: the index of the time series contained (S_k), and the Z-normalized data point values ($X_{N1}^{S_k}, X_{N0}^{S_k}$). Since $X_{N1}^{S_k}$ and $X_{N0}^{S_k}$ are usually floating point values, tuples save significant amount of space without recording the exact normalized values. Moreover, new data can be easily handled by either updating the index element S_k in the corresponding tuples, if their C_1^j's already exists, or else simply creating and adding new tuples.

Example 4.1: We run Algorithm 1 on the data set shown in Table II. Based on different C_1^j's, the data set is first partitioned into two sets: $S_1 = \{1, 3, 4, 5, 6, 8\}, S_2 = \{2, 7\}$. The the Z-normalized data point values are ($X_{N1}^{S_1} = 1.15, X_{N0}^{S_1} = -0.87$) and ($X_{N1}^{S_2} = 1.58, X_{N0}^{S_1} = -0.63$), respectively. Then a prefixed data point set $I_P = \{2, 5\}$ is randomly chosen ($I_Q = \{1, 3, 4, 6, 7\}$), according to which S_1 is further partitioned into two sets: $\{1, 3, 4, 6, 8\}$, and $\{5\}$. With all the information thus far, three tuples are encapsulated as $T_1 = \{\{1, 3, 4, 6, 8\}, 1.15, -0.87\}, T_2 = \{\{2, 7\}, 1.58, -0.63\}, T_3 = \{\{5\}, 1.15, -0.87\}$.

V. Bound Estimation

In this section, we discuss how to devise a tight upper bound of the correlation between a SNP and the query trait to effectively prune the search space. We also formally prove that the proposed bound is tighter than other alternatives.

A. *Upper Bounding using UB_Fab*

Fabrique computes an upper bound UB_k for each of the subset S_k. UB_k includes two parts: the exact correlation for the prefixed subsequences taken from the time series and the query, and the upper bounding function *UB_Fab* proposed in this work. The first part is fixed. The second part determines the tightness of the bound.

UB_Fab aims to maximize the correlation of the two subsequences defined on the remaining data points (by I_Q). From Equation 2, correlation is the sum of inner products after Z-normalization. To get the upper bound, UB_Fab performs the inner product on the two sorted subsequences (both in the same order). This ensures that each term in the product is maximized.

Algorithm 2 describes the bound estimation procedure. First, The query Y is divided into two subsequences, Y^{pre} and Y^{rem}, according to the prefixed data point set I_P and the remaining data point set I_Q, respectively (Line 1, 2). Following this, Y^{rem} is sorted descendingly to form a new subsequence Y^{srt} (Line 3).

Then, for each of T_k (Line 4) obtained from the preprocessing procedure, an arbitrary time series (denoted by the X^j) is picked from the subset (Line 5). Substituting the 1/0 by the corresponding $X_{N1}^{S_k}/X_{N0}^{S_k}$ also encapsulated in this T_k, a Z-normalized X^j is obtained. Its data points corresponding to I_P are then selected, denoted by X^{j_pre}, which are identical for all the binary time series within this T_k if they were Z-normalized and being selected by the I_P. With this X^{j_pre} and Y^{pre} (obtained previously), their exact correlation is calculated (Line 6).

Next, the remaining subsequence for X^j, corresponding to the set I_Q, is sorted in descending order to form X^{j_srt}. In particular, sorting X^{j_srt} can be accomplished by constructing a synthetic sequence, since the length $(N-P)$ and number of 1's in this sequence $(C_1^j - C_1^{j\_pre})$ are both known (Line 7). The correlation between X^{j_srt} and Y^{srt} can be computed as an upper bound (UB_Fab) of the correlation between the subsequence set $X_{\{q\}}^{S_k}$ and the query subsequence $Y_{\{q\}}$ (Line 8).

Finally, the overall upper bound for the correlation of the entire set T_k and the query Y is the summation of the two, calculated in Line 6 and 8, respectively (Line 9). The result of Algorithm 2 is a set of upper bounds for each corresponding subset (Line 11).

Example 5.1: We run Algorithm 2 on the data set shown in Table I., with the tuples produced in Example 4.1., and a query $Y = \{6.07, -2.09, 2.15, -3.67, 0.24, 10.05, 1.02\}$. Take $T_1 = \{\{1,3,4,6,8\}, 1.15, -0.87\}$, and $I_P = \{2,5\}$ for example. $UB_1 = Corr(\{-0.87,-0.87\},\{-2.09, 0.24\}) + UB\_Fab(\{1.15, 1.15, 1.15, -0.87, -0.87\},\ \{6.07, 2.15, -3.67, 10.05, 1.02\}) = Corr(\{-0.87,-0.87\},\{-2.09,0.24\}) + Corr(\{1.15, 1.15, 1.15, -0.87, -0.87\},\ \{10.05, 6.07, 2.15, 1.02, 3.67\}) = 24.93$.

Algorithm 2: Estimation_Fabrique

```
Input: A collection of tuples ${T_k}_{k∈K}$, a query time
       series $Y$ (already Z-normalized), $I_P, I_Q$
Output: A set of upper bounds ${UB_k}_{k∈K}$
1  $Y^{pre} \leftarrow Y_{\{p\}}$, for all $p \in I_P$;
2  $Y^{rem} \leftarrow Y_{\{q\}}$, for all $q \in I_Q$;
3  $Y^{srt} \leftarrow$ sort $Y^{rem}$ descendingly;
4  for $k \leftarrow 1$ to $|K|$ do
5      $X^{j\_pre} \leftarrow \mathrm{X}^j_{\{p\}}$, with $X_{N1}^{S_k}, X_{N0}^{S_k}$, using $\forall\ \mathrm{X}^j \in S_k$, for all $p \in I_P$;
6      Compute $Corr(X^{j\_pre}, Y^{pre})$;
7      $X^{j\_srt} \leftarrow \{\underbrace{X_{N1}^{S_k}, \cdots, X_{N1}^{S_k}}_{C_1^j - C_1^{j\_pre}}, \underbrace{X_{N0}^{S_k}, \cdots, X_{N0}^{S_k}}_{Q-(C_1^j - C_1^{j\_pre})}\}$;
8      **UB_Fab**$(X^{j\_srt}, Y^{srt}) \leftarrow Corr(X^{j\_srt}, Y^{srt})$;
9      $UB_k \leftarrow Corr(X^{j\_pre}, Y^{pre}) +$ **UB_Fab**$(X^{j\_srt}, Y^{srt})$;
10 end
11 return a set of upper bounds ${UB_k}_{k∈K}, |K| \le M$
```

Proposition 5.2: UB_Fab introduces no false dismissals.

Proof: Suppose not. Then for a certain subset S_k, $\exists \mathrm{X}^\epsilon$ s.t. $\mathrm{X}^\epsilon \in S_k$, $Corr(X^\epsilon, Y) > UB_k$. Since $\mathrm{X}_p^\epsilon = \mathrm{X}_p^j$, for all $p \in I_P, \mathrm{X}^j \in S_k \setminus \{\mathrm{X}^\epsilon\}$, the inequality holds because there exists at least a pair of data points $\alpha, \beta \in I_Q$, s.t. (1) $\mathrm{X}_\alpha^\epsilon \neq \mathrm{X}_\alpha^{j\_srt}, \mathrm{X}_\beta^\epsilon \neq \mathrm{X}_\beta^{j\_srt}$, and (2) $\mathrm{X}_\alpha^\epsilon = \mathrm{X}_\beta^{j\_srt}, \mathrm{X}_\beta^\epsilon = \mathrm{X}_\alpha^{j\_srt}$, and (3) $\mathrm{X}_\alpha^\epsilon \neq \mathrm{X}_\beta^\epsilon$.

Without loss of generality, $\mathrm{X}_\alpha^\epsilon = 0$, $\mathrm{X}_\beta^\epsilon = 1$. Therefore,

$$\mathrm{X}_\alpha^{j\_srt} = 1, \mathrm{X}_\beta^{j\_srt} = 0, \text{and} \quad Y_\alpha^{srt} > Y_\beta^{srt}. \tag{4}$$

For simplicity, we write $Y_\alpha = Y_\alpha^{srt}$, and $Y_\beta = Y_\beta^{srt}$.

Hence, $Corr(X^\epsilon, Y) > UB_k$ must satisfy that $(Y_\alpha X_\alpha^\epsilon + Y_\beta X_\beta^\epsilon) > (Y_\alpha X_\alpha^{j\_srt} + Y_\beta X_\beta^{j\_srt})$ (Note the correlation is performed on the Z-normalized data points). Rewriting it by substituting $\mathrm{X}_\alpha^\epsilon = \mathrm{X}_\beta^{j\_srt}, \mathrm{X}_\beta^\epsilon = \mathrm{X}_\alpha^{j\_srt}$, the inequality then becomes

$$(Y_\alpha X_\alpha^\epsilon + Y_\beta X_\beta^\epsilon) - (Y_\alpha X_\beta^\epsilon + Y_\beta X_\alpha^\epsilon) > 0. \tag{5}$$

The left-hand side of Equation 5 = $(Y_\alpha - Y_\beta)(X_\alpha^\epsilon - X_\beta^\epsilon)$. The first term $(Y_\alpha - Y_\beta) > 0$ because of Equation 4. The second term $(X_\alpha^\epsilon - X_\beta^\epsilon) < 0$ because of the assumption preceding Equation 4. The inequality in Equation 5 thus should not hold. ■

B. *Comparing with LB_Keogh*

LB_Keogh [15], [16], [21] is a bounding method adopted by many works [25]. The measure metric supported by

LB_Keogh includes DTW, and Euclidean distance. The latter is defined as,

$$EuD(X^j, Y) = \sqrt{\sum_{i=1}^{N} (X_i^j - Y_i)^2}. \qquad (6)$$

Given a query Y and a set of normalized time series $S_k = \{X_i^j\}, 1 \leq i \leq N, 1 \leq j \leq |S_k|$, LB_Keogh computes the lower bound of their Euclidean distance (squared),

$$LB\_Keogh(S_k, Y) = \sum_{i=1}^{N} \begin{cases} (Y_i - U_i)^2 & \text{if } Y_i > U_i \\ (Y_i - L_i)^2 & \text{if } Y_i < L_i \\ 0 & \text{otherwise.} \end{cases}$$

The new sequences Upper (U) and Lower (L) are defined as,

$$U_i = \max(X_i^1, \cdots, X_i^j, \cdots, X_i^{|S_k|})$$
$$L_i = \min(X_i^1, \cdots, X_i^j, \cdots, X_i^{|S_k|}).$$

UB_Fab is an upper bounding method based on correlation, while LB_Keogh is used for lower bounding using Euclidean distance. Nevertheless, comparing the tightness of these two bounding techniques is still possible. Previous work [26], [27] has demonstrated that, correlation and squared Euclidean distance are equivalent to measure the similarity of a query and the time series, provided they are both Z-normalized,

$$Corr(X^j, Y) = 1 - \frac{1}{2N} EuD^2(X^j, Y). \qquad (7)$$

By Equation 7, UB_Fab can be transformed into its equivalent lower-bounding form LB_Fab_{EuD}, using Eucliean distance. In this way, maximum correlation is equivalent to minimum Euclidean distance (squared) in the nearest-neighbor query.

Therefore, the problem of comparing the tightness of UB_Fab and LB_Keogh is equivalent to consider the dual problem, which is comparing LB_Fab_{EuD} and LB_Keogh.

Proposition 5.3: LB_Fab_{EuD} builds tighter lower bound than LB_Keogh.

Proof: To prove the proposition is to prove

$$LB\_Keogh \leq LB\_Fab_{EuD},$$

where

$$\begin{aligned} LB\_Fab_{EuD} &= EuD^2(X^{j\_srt}, Y^{srt}) \\ &= \sum_{i=1}^{N} (X_i^{j\_srt} - Y_i^{srt})^2, \quad \forall X^j \in S_k. \end{aligned}$$

For a set of binary time series S_k (with the same C_1^j's), the Upper and Lower are,

$$U_i = \max(X_i^1, \cdots, X_i^j, \cdots, X_i^{|S_k|}) = X_{N1}^j = U'$$
$$L_i = \min(X_i^1, \cdots, X_i^j, \cdots, X_i^{|S_k|}) = X_{N0}^j = L'.$$
$$X_i^{j\_srt} = X_{N0}^j \quad \text{or} \quad X_{N1}^j.$$

There are three cases to consider,

I. $Y_i > U'$ In this case, for LB_Keogh, it computes as $(Y_i - U')^2$. For LB_Fab_{EuD}, it is either $(Y_i - U')^2$ or $(Y_i - L')^2$. As $|Y_i - U'| < |(Y_i - U') + (U' - L')| = |Y_i - L'|$, $(Y_i - U')^2 \leq (Y_i - L')^2$. Hence $LB\_Keogh \leq LB\_Fab_{EuD}$.

II. $Y_i < L'$ Similar to I., it yields to the same result.

III. $L' \leq Y_i \leq U'$ In this case, for LB_Keogh, it computes as 0. For LB_Fab_{EuD}, it is either $(Y_i - U')^2$ or $(Y_i - L')^2$. Hence $0 = LB\_Keogh \leq LB\_Fab_{EuD}$.

Thus we have shown that $LB\_Keogh \leq LB\_Fab_{EuD}$ ■

C. Comparing with UB_EBC

Bounding methods have also been studied in many other applications such as template matching, an important problem in imagine processing. Given a template Y, the task is to identify the best-matched window, denoted by X^{j*}, for an input image represented by $\{\mathrm{X}^j\}_{j \in \{1,2,\cdots,M\}}$ as M windows of the equal size as Y. This can be defined as

$$Corr(X^{j*}, Y) = \arg\max_j Corr(X^j, Y).$$

The images for both X^j's and Y are regarded as time series because they are vectorized.

One notable work on template matching proposed an algorithm called EBC [17], which uses upper bounding functions to skip candidate windows (i.e., the X^j's). The core part of EBC is the upper bounding functions UB_EBC. Specifically, for an arbitrary subsequences of X^j and Y, indexed by an index set $I_R \subset \{1, 2, \cdots, N\}$ and denoted by $\{\mathrm{X}_r^j\}_{r \in I_R}$ and $\{Y_r\}_{r \in I_R}$, respectively, UB_EBC upper bounds the cross correlation of such two subsequences,

$$\sum_{r=1}^{|I_R|} \mathrm{X}_r^j Y_r \leq \|\{X_r^j\}\| \, \|\{Y_r\}\|,$$

which holds because of the Cauchy-Schwarz inequality

Proposition 5.4: UB_Fab is tighter than UB_EBC.

Proof: For simplicity, we write

$$\mathbf{a} = \{\mathrm{X}_r^j\}_{r \in I_R}, \mathbf{b} = \{Y_r\}_{r \in I_R}.$$

To prove Proposition 5.4., we need to prove,

$$\sum_{r=1}^{|I_R|} a_r b_r \leq UB\_Fab(a^{srt}, b^{srt}) \leq \|\mathbf{a}\| \, \|\mathbf{b}\| \qquad (8)$$

The first inequality in Equation 8 holds because of Proposition 5.2. The second holds since a sorted vector does not change its l_2-norm. ■

VI. Nearest-Neighbor Query

This section outlines the proposed Fabrique framework for identifying the nearest-neighbor of a given query. Its time complexity is also analyzed.

A. Main Algorithm

Algorithm 3 leverages Fabrique for fast query. The algorithm begins by invoking the two procedures in Fabrique, i.e., **preprocessing** (Line 1) and **estimation** (Line 2). The resulting UB_k's are sorted in descending order (Line 3). Next, each subset S_k encapsulated in a corresponding tuple T_k is visited by examining all the binary time series contained (Line 5). The maximum correlation between this set S_k and a query Y is thereby identified (Line 6). It is then compares with the maximum correlation recorded so far (max-corr-so-far, Line 7), to determine if the max-corr-so-far needs to be updated (Line 8). The program terminates when max-corr-so-far is larger or equal than the upper bound UB_{k+1} of the next subset S_{k+1} in the sorted order (Line 9-12).

Algorithm 3: Nearest-Neighbor Query

```
Input: A set of time series
       $\{X_i^j\}, 1 \le i \le N, 1 \le j \le M$, a query $Y$
Output: A time series $X^{j*}$ such that $Corr(X^{j*}, Y) \ge$
       $Corr(X^j, Y), \forall\ j \in \{1, 2, \cdots, M\} \setminus \{j*\}$
1  $\{T_k\}_{k \in K}, I_P, I_Q$ = Preprocessing_Fabrique($\{X_i^j\}$);
2  $\{UB_k\}_{k \in K}$ = Estimation_Fabrique($\{T_k\}_{k \in K}, I_P, I_Q$);
3  $\{UB_k\}_{k \in K} \leftarrow$ Sort $\{UB_k\}_{k \in K}$ descendingly;
4  max-corr-so-far $\leftarrow 0$;
5  for $k \leftarrow 1$ to $|K|$ do
6      currentCorr = max$\{Corr(X^j, Y), \forall j \in S_k\}$;
7      if currentCorr $\ge$ max-corr-so-far then
8          max-corr-so-far $\leftarrow$ currentCorr;
9          if max-corr-so-far $\ge UB_{k+1}$ then
10             $j* \leftarrow j$ s.t. max-corr-so-far =
               $Corr(X^{j*}, Y)$;
11             break;
12         end
13     end
14 end
15 return $X^{j*}$
```

Search space is pruned by employing the upper bounding function UB_Fab. When Algorithm 3 terminates after examining all the time series in S_{k*}, the max-corr-so-far is larger or equal than the upper bound of the next subset in the sorted descending order. All the binary time series in the remaining and un-examined subsets (from S_{k*+1} to $S_{|K|}$) are pruned from the further analysis, since their correlation with the query cannot exceed the upper bound UB_{k*+1}.

Computing the correlation of a time series X^j from a subset S_k and a query Y requires first substituting its 1/0 with the corresponding $X_{N1}^{S_k}$ / $X_{N0}^{S_k}$, utilizing the data structure of tuples. The substitution and the follow-up correlation computation can be saved once the upper bound of a subset is smaller than max-corr-so-far. This explains the reason for postponing substitution to a late stage and the possibility that it may eventually be saved, as mentioned in Section IV Part A.

B. Complexity Analysis

The brute-force approach (BFA) is similar to Algorithm 3, except (1) the preprocessing part consists of Z-normalization only, (2) it needs no bound estimation, and (3) it computes the correlations between a query and the entire set of time series exhaustively to find the maximal among them.

Preprocessing_Fabrique takes $O(N \cdot M + N)$ for Z-normalization, $O(M)$ for partitioning, accumulating to $O(N \cdot M + N + M)$. Z-normalization in BFA is $O(N \cdot M + N \cdot M + N \cdot M)$ for the three-step routine but no partitioning is further required. Given that $N \ll M$, the overhead of BFA so far is already significant.

Estimation_Fabrique takes $O(Q \cdot logQ + |K|)$ for bound estimation, where $O(Q \cdot logQ)$ is the cost for sorting Y^{rem} into Y^{srt}, and $Q < N$, $|K| < M$.

For the rest part, Fabrique takes $O(M)$ for computing the correlations in the worst case, and another $O(M)$ for maintaining the max-corr-so-far. The complexity for BFA is also $O(M + M)$.

To sum up, Fabrique takes $O(N \cdot M + N + M + Q \cdot logQ + |K| + 2M)$, as compared with $O(N \cdot M + N \cdot M + N \cdot M + 2M)$ from BFA. Given that M is the dominant factor, the overhead of BFA is significant over Fabrique.

VII. Experimental Evaluation

In this section, the performance of Fabrique is first evaluated in quantitative trait association study. Its two components, **Preprocessing_Fabrique** and **Estimation_Fabrique** procedures, are then separately applied for memory usage optimization and template matching, respectively. We aim to demonstrate that both the entire framework of Fabrique and its individual components can be adapted to a wide range of real-life problems dealing with binary time series. We also show that the upper bound developed in Fabrique can be applied to template matching and achieves better performance than alternative method.

A. Quantitative Trait Association Study

The SNP data set used for this experiment is a benchmark data set downloaded from the Jackson Lab (http://www.jax.org/). The SNPs whose C_1^j's are less than 5% of the number of samples are removed from analysis. The final data set under study contains 64,973 SNPs ($M = 64,973$), one real phenotype (neurosensory), and the number of samples is 34 ($N = 34$). The program is implemented in Matlab, Version 2012a. The experiments are performed on a 2.9 GHz Intel Core i7 MacBook Pro with 8 GB 1600 MHz DDR3 memory running OS X Version 10.8.3.

Fabrique is the first work that explicitly models quantitative trait association study in terms of time-series query problem. For comparison, we select the state-of-the-art method, Matrix eQTL [14], from the bioinformatic community, which is specifically designed to solve the quantitative trait association study problem. It has been shown that Matrix eQTL is orders of magnitudes faster than other exiting approaches.

1) Identifying the Most Significant SNP: Figure 2 shows the (a) runtime and (b) pruning ratio comparison of Fabrique and Matrix eQTL. Fabrique takes less than half of the time as Matrix eQTL does to complete the task. It also achieves high pruning ratio, comparing

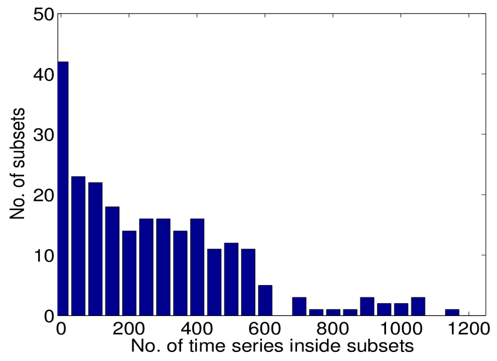
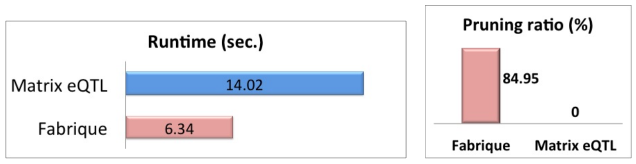


Fig. 1. Distribution of the sizes of 237 subsets from partitioning a data set with 64,973 time series, using a prefixed data point set I_P, where P =4.



(a) Runtime (b) Pruning ratio

Fig. 2. Performances comparison between Fabrique and Matrix eQTL for identifying most significant SNPs for each one of the 1000 phenotypes (1 real + 999 artificial). The parameters in the Fabrique framework are set as $P = 10, \theta = 200$.

with none from Matrix eQTL. We tried to preserve the downloaded source code of Matrix eQTL (Available at http://www.bios.unc.edu/research/genomic_software/Matrix _eQTL/) as original as possible throughout the experiments.

Matrix eQTL explicitly computes the correlation for every possible SNP-trait pair. A major part of Matrix eQTL mainly leverages matrix multiplication to improve the efficiency. The resulting speedup heavily relies on the software (R or Matlab, in different versions) in which the program is implemented, and the hardware configuration (number of cores, main memory, CPU/GPU clock, etc). On the other hand, the pruning ratio from Fabrique assures the potential for studying eQTL problems in large scale.

Tuning the two parameters (P and θ) further reduces the overall runtime. As the parameter controlling the number of prefixed data points, P impacts the computation costs of the components in the Fabrique framework for Z-normalization, partitioning, estimation and sorting of the upper bounds, and finally identifying the most correlated SNPs. Fig. 3 provides a case study of how parameter P affects both the overall runtime and each of its components. With the increasing of P, the time spent on partitioning, bound estimation and sorting increases, while the time for actual finding the SNPs decreases. One extreme case is that, if P is set to be nearly as large as N (the length of SNPs), the time for SNP identification will virtually decrease to zero, as almost every single SNP constitutes a subset itself. Such configuration, however, should not be adopted, because it increases the time of other components and the overall runtime.

The other parameter θ determines the number of subsets to be examined at the same time. Figure 4 demonstrates that the runtime is not a monotonic function of the merging parameter

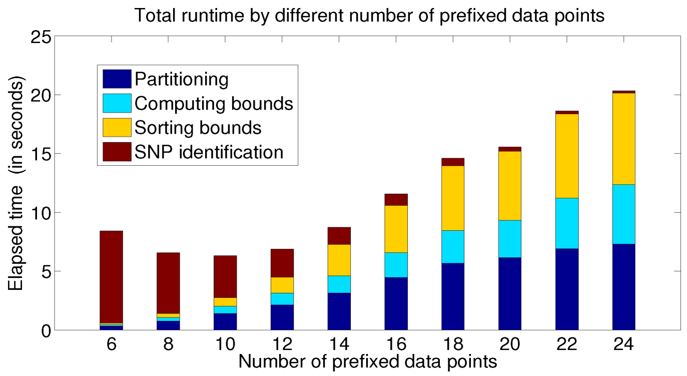


Fig. 3. Runtime analysis for each of the components in the Fabrique framework when varying the parameter of P. The time spent on Z-normalization using **Proprecessing_Fabrique** is too tiny to be shown in this plot and hence omitted.

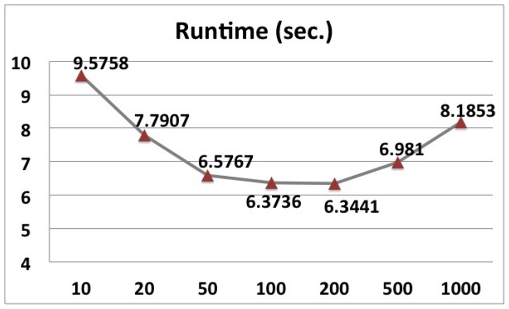


Fig. 4. Runtime analysis when varying the parameter of θ from 10 to 1000.

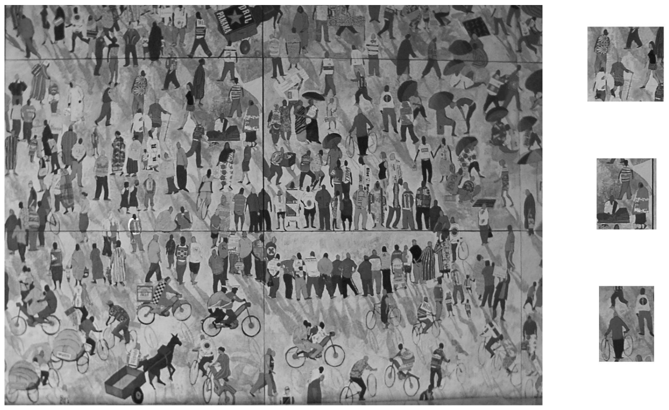
θ. The underlying reason is that the pruning ratio deceases with the increase of θ. The sacrifice of a certain amount of pruning ratio is only worthwhile when θ is moderately configurated.

2) Permutation Tests: In genetic association studies, permutation tests are usually employed to assess the statistical significance of the identified SNP-trait pairs. For a given trait, the max correlation obtained from it and the SNP identified is recorded as the maximum test statistic. The trait is then permuted a number of times and the above process is repeated to find the maximum test statistic for each permuted trait. The significance is the proportion of permuted traits whose maximum test statistics are greater than the correlation of the identified SNP-trait pair using original trait.

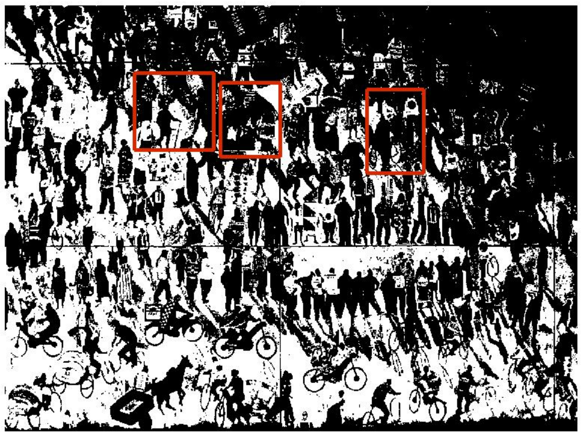
In our experiments, the trait is permuted 1000 times. During the test, for every permuted trait, the program terminates whenever the correlation from a SNP-trait (permuted) exceeds the one from the SNP-trait (real). The experiment results are shown in Fig. 5. Fabrique is over 13X faster than Matrix eQTL and achieves high pruning ratio.

B. Memory Usage Optimization

The **Preprocessing_Fabrique** procedure in the framework provides a fast approach to perform Z-normalization on the binary time series. **Preprocessing_Fabrique** also suggests a promising approach to occupy the minimum space, either when loading the Z-normalized binary time series into the memory, or storing them on disks or drives. We set P to 6, and still use the SNP data set in the previous section. The Z-normalized data using the tuple data structure takes about 352K bytes, while



(a) Original data in grayscale



(b) Matched results in binary setting

Fig. 6. Template matching results on data set *Paint*.The best-matched locations are identified and highlighted by red rectangles, showing the correctness of applying the bounding technique UB_Fab in **Estimation_Fabrique**.

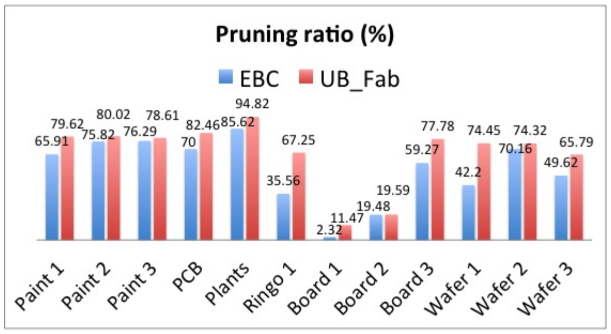


Fig. 7. Performance comparison between UB_EBC and UB_Fab for pruning ratio using the same experiment setting from the EBC paper.

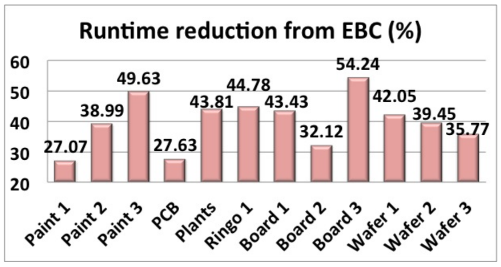
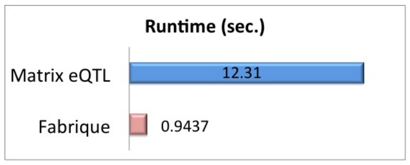
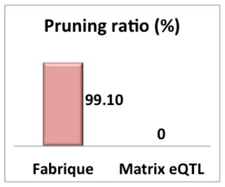


Fig. 8. Runtime reduction from using **Estimatin_Fabrique** procedure over the EBC algorithm for each of the data set experimented.



(a) Runtime



(b) Pruning ratio

Fig. 5. Performances comparison between Fabrique and Matrix eQTL using permutation tests. The parameters in the Fabrique framework are set as $P = 2, \theta = 11$.

the plain form recorded using double precision takes 17.67M bytes. Thus **Preprocessing_Fabrique** saves more than 50X of the space while loosing no information.

C. *Template Matching*

We also apply our upper bound to binary template matching problem and compare it with UB_EBC in [17] to evaluate its performance.

To make a fair comparison, we implemented the EBC algorithm according to the pseudo-code in [17], and only substituted the UB_EBC routine for UB_Fab in the procedure to acquire the upper bounds. In addition, we adopted all the parameters best-tuned for UB_EBC , and the exact same data (Available at http://www.vision.deis.unibo.it/smatt/Pattern Matching.html). One example problem is demonstrated in Fig 6. Fig 7 shows that UB_Fab always achieves higher pruning ratio than UB_EBC, which further evidences Proposition 5.4 in practice.

Besides the improved pruning ratio, employing the UB_Fab in the **Estimation_Fabrique** procedure for upper-bound estimation require much less computation load than the l_2-norm calculation in UB_EBC does. Fig. 8 shows that the **Estimatin_Fabrique** procedure reduces the runtime from at least 27% in the worst case, and up to over 50% in the best case.

VIII. Conclusion

This paper introduces a framework Fabrique to tackle the problem of intensive computation burden existing in quantitative trait association study. Fabrique formulates the problem in terms of time-series query. Several techniques, such as partitioning, Z-normalization, and upper bound estimation, are developed to support fast identification of significant SNP-trait pairs. Experimental results demonstrate that Fabrique outperforms the state-of-the-art work and achieves high pruning ratio. Although the motivation for developing Fabrique is to study genetics problems, it also provides a recipe for other real-life applications, such as template matching.

IX. Acknowledgements

The work was partially supported by the US National Science Foundation grants IIS-1162374 and IIS-1218036.

References

[1] The 1000 Genomes Project Consortium, "A map of human genome variation from population-scale sequencing," *Nature*, vol. 467, pp. 1061–1073, 2010.

[2] B. Li and S. Leal, "Methods for detecting associations with rare variants for common diseases: Application to analysis of sequence data," *Am J Hum Genet*, vol. 83(3), pp. 311–321, 2008.

[3] B. R. Bochner, "New technologies to assess genotype phenotype relationships," *Nature Reviews Genetics*, vol. 4, pp. 309–314, 2003.

[4] M. V. Rockman and L. Kruglyak, "Genetics of global gene expression," *Nature Reviews Genetics*, vol. 7, pp. 862–872, 2006.

[5] J. Michaelson, S. Loguercio, and A. Beyer, "Detection and interpretation of expression quantitative trait loci (eQTL)," *Methods*, vol. 48(3), pp. 265–276, 2009.

[6] E. E. Schadt, C. Molony, E. Chudin, K. Hao, X. Yang, P. Y. Lum, A. Kasarskis, B. Zhang, S. Wang, C. Suver, J. Zhu, J. Millstein, S. Sieberts, J. Lamb, D. GuhaThakurta, J. Derry, J. D. Storey, I. Avila-Campillo, M. J. Kruger, J. M. Johnson, C. A. Rohl, A. van Nas, M. Mehrabian, T. A. Drake, A. J. Lusis, R. C. Smith, F. P. Guengerich, S. C. Strom, E. Schuetz, T. H. Rushmore, and R. Ulrich, "Mapping the genetic architecture of gene expression in human liver," *PLoS biology*, vol. 6(5), p. e107, 2008.

[7] M. Pagano and K. Gauvreau, *Principles of Biostatistics*. Pacific Grove, CA: Duxbury Press, 2000.

[8] R. G. Miller, *Simultaneous Statistical Inference*. Springer Verlag New York, 1981.

[9] G. A. Churchill and R. W. Doerge, "Empirical threshold values for quantitative trait mapping," *Genetics*, vol. 138(3), pp. 963–971, 1994.

[10] P. McClurg, J. Janes, C. Wu, D. Delano, J. Walker, S. Batalov, J. Takahashi, K. Shimomura, A. Kohsaka, J. Bass, T. Wiltshire, and A. Su, "Genomewide association analysis in diverse inbred mice: Power and population structure," *Genetics*, vol. 176(1), pp. 675–683, 2007.

[11] S. Purcell, B. Neale, K. Todd-Brown, L. Thomas, M. Ferreira, D. Bender, J. Maller, P. Sklar, P. de Bakker, M. Daly, and P. Sham, "Plink: a tool set for whole-genome association and population-based linkage analyses," *Am J Hum Genet*, vol. 81(3), pp. 559–575, 2007.

[12] D. Leung, "An r package for analysis of whole-genome association studies," *Hum. Hered.*, vol. 64, pp. 45–51, 2007.

[13] D. M. Gatti, A. A. Shabalin, T.-C. Lam, F. A. Wright, I. Rusyn, and A. B. Nobel, "Fastmap: Fast eqtl mapping in homozygous populations," *Bioinformatics*, vol. 25(4), pp. 482–489, 2009.

[14] A. A. Shabalin, "Matrix eqtl: ultra fast eqtl analysis via large matrix operations," *Bioinformatics*, vol. 28, no. 10, pp. 1353–1358, 2012.

[15] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[16] E. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos, "LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures (expanded version)," *VLDB*, 2006.

[17] S. Mattoccia, F. Tombari, and L. D. Stefano, "Fast full-search equivalent template matching by enhanced bounded correlation," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 528–538, 2008.

[18] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *PVLDB*, 2008.

[19] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pp. 69–84, 1993.

[20] E. Keogh, K. Chakrabarti, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *Proceedings of ACM SIGMOD Conference on Management of Data*, pp. 151–162, 2001.

[21] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*, pp. 262–270, 2012.

[22] D. Wu, D. Liu, Z. Puskas, C. Lu, A. Wimmer, C. Tietjen, G. Soza, and S. Zhou, "A learning based deformable template matching method for automatic rib centerline extraction and labeling in ct images," *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 980–987, 2012.

[23] Y. Hel-Or, H. Hel-Or, and E. David, "Fast template matching in non-linear tone-mapped images," *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 1355–1362, 2011.

[24] M. Mori and K. Kashino, "Fast template matching based on normalized cross correlation using adaptive block partitioning and initial threshold estimation," *2010 IEEE International Symposium on Multimedia*, pp. 196–203, 2010.

[25] "LB_Keogh homepage," http://www.cs.ucr.edu/%7Eeamonn/LB_Keogh.htm.

[26] S. Borgatti, Distance and Correlation. [Online]. Available: http://www.analytictech.com/mb876/handouts/distance_and_correlation.htm

[27] D. Rafiei, "On similarity-based queries for time series data," *IEEE 25th International Conference on Data Engineering, (ICDE)*, pp. 410–417, 1999.