A generic algorithm to find all common intervals of two permutations

Guiliang Feng
Center of Advanced Computer Study
University of Louisiana at Lafayette
Lafayette, LA 70504
glf@cacs.louisiana.edu

Yujiang Shan
Department of Computer Science
Southern Arkansas University
Magnolia, AR 71753
yshan@saumag.edu

Abstract

Let $\aleph$ be the set of $\{1, 2, , ..., m\}$, $[x, y]$ denote the set of $[x, x + 1, ..., y]$, where $1 \leq x, y \leq m$. Given two permutations σ_A and σ_B of a set $\aleph$, A 2-tuple of intervals $([x_1, y_1], [x_2, y_2])$ is called common intervals if $\sigma_A([x_1, y_1]) = \sigma_A([x_2, y_2])$. In this paper, we propose a sufficient and necessary condition for a 2-tuple of intervals to be common intervals. Based on these conditions, we present a generic algorithm that finds all common intervals of these two permutations.

1 Introduction

The problem of finding common intervals has been drawing much attention in recent years as a useful tool in the study of comparative genomics. The notion of common intervals can be used to detect possible evolutional associations between genes ([2]) and functional connection between proteins ([4], [5]).

Uno and Yagiura [1] presented an optimal $O(n + K)$ time algorithm for finding all common intervals where K ($\leq \binom{n}{2}$) is the number of outputs. This algorithm is further extended in [3] to find all common intervals of k permutations in optimal $O(nk + K)$ time.

However, the success of the algorithm in [1] relies on one of two conditions (i.e. Lemma 4.1 and Lemma 4.2 in [1]) on the permutations. In this paper, we present a sufficient and necessary condition for a 2-tuple of intervals to be common intervals. Based on these conditions, an algorithm to find all common intervals of the permutations is proposed. The algorithm requires no condition, thus can be applied for any two permutations.

For a permutation π and an interval $[x, y]$, following functions are defined: $l(x, y) = \min_{i \in [x,y]} \pi(i)$, $u(x, y) = \max_{i \in [x,y]} \pi(i)$, $f(x, y) = u(x, y) - l(x, y) - (y - x)$.

Two basic lemmas are given below:

Lemma 1.1. *Let π be a permutation of set $\aleph = \{1, 2, 3, ..., m\}$, x, α and β be elements in $\aleph$ and $\alpha < x < \beta$,*

(i) If $\pi(x) > \pi(\alpha) > \pi(\beta)$, then $f(x, z) > 0$ for every z, $\beta \leq z \leq m$;

(ii) If $\pi(x) < \pi(\alpha) < \pi(\beta)$, then $f(x, z) > 0$ for every z, $\beta \leq z \leq m$;

(iii) If $\pi(x) > \pi(\beta) > \pi(\alpha)$, then $f(z, x) > 0$ for every z, $1 \leq z \leq \alpha$;

(iv) If $\pi(x) < \pi(\beta) < \pi(\alpha)$, then $f(z, x) > 0$ for every z, $1 \leq z \leq \alpha$.

Lemma 1.2. *If $[a, b]$ is not a common interval, then either (i) or (ii) holds:*

(i) There exists an element $t_0 \in [a, b]$ such that $f(a, z) > 0$ for every z, $t_0 \leq z \leq m$.

(ii) There exists an element $t_0 \in [a, b]$ such that $f(z, b) > 0$ for every z, $1 \leq z \leq t_0$.

2 Preliminary Algorithm

The input of the algorithm is a permutation on $\aleph = \{1, 2, 3, ..., m\}$ which can be represented as a one dimensional array $\pi[m]$:

$$\begin{pmatrix} 1 & 2 & \dots & x & \dots & m \\ \pi(1) & \pi(2) & \dots & \pi(x) & \dots & \pi(m) \end{pmatrix} \tag{2.1}$$

Let $a_{ij} = f(i, j)$, $1 \leq i \leq m - 1$ and $2 \leq j \leq m$. $a_{ij} = 0$ if and only if $[i, j]$ is a common interval. Lemma 1.1 enables us to identify intervals that are not common intervals without calculation; Lemma 1.2 guarantees that those remaining intervals are common intervals.

Considering the matrix formed by a_{ij}, the main idea of the algorithm is to determine the lower bound in each row and upper bound in each column of the entries that are not common intervals

For each element x in (2.1), we define $UP(x) = \{a|a < x$ and $\pi(a) > \pi(x)\}$, $US(x) = \{b|b > x$ and $\pi(b) > \pi(x)\}$, $LP(x) = \{c|c < x$ and $\pi(c) < \pi(x)\}$, $LS(x) = \{d > x$ and $\pi(d) < \pi(x)\}$. Note: (1) Each of these sets could be empty and will be denoted as ϕ; (2) $UP(x) \cup LP(x) = [1, x-1]$ and $US(x) \cup LS(x) = [x+1, m]$.

Following steps are followed to find the horizontal boundary y_x:

Step 1. Find $i \in LP(x)$ that $\pi(i) = max\{\pi(LP(x))\}$.

Step 2. Find μ, the minimum value of $LS(x)$ such that $\pi(i) > \pi(\mu)$.
If either $LP(x)$ or $LS(x)$ is empty; or if neither $LP(x)$ nor $LS(x)$ is empty but no such μ exists, we define $\mu = \infty$.

Step 3. Find $j \in UP(x)$ that $\pi(j) = min\{\pi(UP(x))\}$.

Step 4. Find λ, the minimum value of $US(x)$ such that $\pi(j) < \pi(\lambda)$.
If either $UP(x)$ or $US(x)$ is empty; or if neither $UP(x)$ nor $US(x)$ is empty but no such λ exists, we define $\lambda = \infty$.

Step 5. Take $y_x = min\{\mu, \lambda\}$.

Similar steps can be followed to find the vertical boundary z_x:

Step 1. Find $h \in LS(x)$ that $\pi(h) = max\{\pi(LS(x))\}$.

Step 2. Find α, the maximum value of $LP(x)$ such that $\pi(\alpha) < \pi(h)$.
If either $LP(x)$ or $LS(x)$ is empty; or if neither $LP(x)$ nor $LS(x)$ is empty but no such α exists, we define $\alpha = 0$.

Step 3. Find $k \in US(x)$ that $\pi(k) = min\{\pi(US(x))\}$.

Step 4. Find β, the maximum value of $UP(x)$ such that $\pi(\beta) > \pi(k)$.
If either $UP(x)$ or $US(x)$ is empty; or if neither $UP(x)$ nor $US(x)$ is empty but no such β exists, we define $\beta = 0$.

Step 5. Take $z_x = max\{\alpha, \beta\}$.

Finally, by the procedure below, all the common intervals are determined:

```
for( i = 2; i ≤ m − 1; i++)
    if (z_i < i − 1)
        for( j = z_i + 1; j ≤ i − 1; j++)
            if( y_j > j )
                output common interval [j, i]
```

3 Implementation of the Preliminary Algorithm

The major steps of the preliminary algorithm described in the previous section include computation of following values for each x, $2 \leq x \leq m-1$:

(i) $\max\{\pi(LP(x))\}$, $\max\{\pi(LS(x))\}$, $\min\{\pi(UP(x))\}$ and $\min\{\pi(US(x))\}$;

(ii) $\lambda(x)$, $\beta(x)$, $\alpha(x)$ and $\mu(x)$.

In this section, iterative methods for calculating values in (i) and (ii) are presented respectively.

The implementation of the algorithm is devised in such a way that one result is determined using the previous ones, thus all the common intervals can be found in nearly linear time.

References

[1] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. Algorithmica, 26(2):290-309, 2000.

[2] Thomas Schmidt and Jans Stoye. Quadratic Time Algorithms for Finding Common Intervals in Two and More Sequences. Proceedings of CPM 2004, LNCS 3109, 347-359, 2004.

[3] S. Heber and J. Stoye. Finding All Common Intervals of k Permutations. In Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching, CPM 2001, page 207-218, 2001.

[4] E. M. Marcotte, M. Pellegrini, H. L. Ng, D. W. Rice, T. O. Yeates, and D. Eisenberg. Detecting protein function and protein-protein interactions from genome sequences. Science, 285:751-753, 1999.

[5] R. Overbeek, M. Fonstein, M. D'Souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. Proc. Natl. Acad. Sci. USA, 96(6):2896-2901, 1999