8. Common Intervals of Permutations

Literature:

Heber, S., Mayr, R., & Stoye, J. (2009). *Common Intervals of Multiple Permutations.* Algorithmica, 60(2), 175–206.

Bergeron, A., Chauve, C., de Montgolfier, F., & Raffinot, M. (2008). *Computing common intervals of K permutations, with applications to modular decomposition of graphs.* SIAM Journal on Discrete Mathematics, 22(3), 1022–1039.

Rusu, I. (2014). *MinMax-profiles: A unifying view of common intervals, nested common intervals and conserved intervals of K permutations.* Theoretical Computer Science, 543(C), 90–111.

8.1. Gene Clusters

Gene clusters are sets of genes with associated functions, such as genes encoding for different enzymes in a metabolic pathway. Frequently, gene clusters are located close to each other on the genome sequence. Such gene clusters are common in prokaryotes and fungi, but can also be found in higher eukaryotes, even in humans.

The function of the large majority of genes in sequenced organisms remains unknown until today. Yet, without the knowledge of the genes' function, it seems hopeless to identify gene clusters. However, comparative genomics provides a solution, if gene clusters are conserved among related species (or strains with largely diverging gene order): Gene cluster candidates can be identified by finding those sets of genes that are closely located to each other in multiple genomes.

8.2. Common Intervals

Given two permutations π and σ of size n, a pair of intervals $[i, j]$ in π and $[k, l]$ in σ, $1 \leq i < j \leq n$ and $1 \leq k < l \leq n$, are *common intervals* if the set of elements of $\pi[i, j]$ and $\sigma[k, l]$ are identical.

Problem 7. *Given two permutations π and σ of size n. Enumerate all common intervals of π and σ.*

Without loss of generality, assume from now on that $\pi = \mathbf{id}$. Then $\pi[i, j] = [i, j]$, and the task of identifying common intervals is reduced to the study of the order of elements in σ. In doing so, we will say that an interval $[i, j] := i, i + 1, \ldots j$ (i.e., an

interval of **id**) is a *common interval* iff there exists an interval $[k, l]$ in σ such that $\{x \mid x \in \sigma[k, l]\} = \{i, \ldots, j\}$.

Observation 15. *If $[i, j]$ is a common interval, then the smallest and largest index of its elements in σ are not farther apart from each other than $j - i + 1$.*

This observation leads to the following test function $f(i, j)$:

$$f(i,j) := \max(\sigma^{-1}[i,j]) - \min(\sigma^{-1}[i,j]) + 1 - (j - i + 1) \tag{8.1}$$
$$= \max(\sigma^{-1}[i,j]) - \min(\sigma^{-1}[i,j]) - j + i \tag{8.2}$$

Observation 16. *The roots (a.k.a.* zeros*) of $f(i, j)$ are the common intervals of π and σ.*

Consider a naive algorithm for the enumeration of common intervals that tests all possible intervals $[i, j]$ using $f(i, j)$. Its running time would be as follows: the time to identify the maximum and minimum value of an interval is linear in its size. Because at least half of the intervals of σ are larger than or equal to $\frac{n}{2}$, $f(i, j)$ is in $O(n)$. Also, a permutation of size n has $\binom{n}{2}$ intervals, each of which could be a common interval (e.g. $\pi = \sigma$). Consequently, the overall running time of the naive algorithm is $O(n^3)$. Through a careful choice of the succession of tested intervals in σ^{-1} and *updating* (rather than completely re-computing) the largest and smallest elements of successively tested intervals, the computation time for $f(i, j)$ can be amortized. This leads to an improved overall running time of $O(n^2)$ for a more sophisticated algorithm.

However, the number of common intervals of two random permutations is in $O(1)$. An ideal algorithm for enumerating common intervals would not afford more (asymptotic) time than necessary for the reading of the input permutations and the reporting of all common intervals. Such an algorithm is called *efficient* and would have $O(n+z)$ running time for enumerating the $O(z)$ common intervals of two permutations of size n.

8.3. Generators of Common Intervals

Observe that most common intervals are composed of smaller, overlapping intervals: A set of intervals $C = \{c_1, c_2, \ldots, c_m\}$ is said to *generate* interval $[x, y]$ iff $[x, y] = c_1 \cup c_2 \cup \cdots \cup c_m$ and $c_i \cap c_{i+1} \neq \emptyset$ for all $1 \leq i < m$.

Definition 28. *Let $C_{\pi,\sigma}$ be the set of common intervals of π and σ. A common interval c is called* reducible *iff there is a set of common intervals $C' \subseteq C_{\pi,\sigma}$ such that C' generates c and each $d \in C'$ is a proper subset of c. Otherwise, c is called* irreducible.

The set of irreducible intervals of two permutations π and σ is *unique* and subsequently denoted as $I_{\pi,\sigma}$.

Theorem 8. *The size of $I_{\pi,\sigma}$ is within $[1, n - 1]$.*

Proof. There is a surjective mapping between the pairs of elements $(e, e+1)$, $1 \leq e \leq n-1$ and the set of irreducible intervals. More precisely, each pair of elements $(e, e+1)$ is associated with exactly one irreducible interval that contains it. Consider the total order on the set of common intervals of permutation σ given by $(i_1..j_1) < (i_2..j_2)$ iff either $i_1 > i_2$, or $i_1 = i_2$ and $j_1 < j_2$. For each e with $1 \leq e \leq n-1$, let $Small(e)$ denote the smallest, with respect to this order, common interval of σ containing e and $e+1$. Then the set of irreducible intervals of σ is the set $\{Small(e) \mid 1 \leq e \leq n-1\}$. □

The concept of common intervals that generate other intervals can be generalized:

Definition 29. *A* generator *of common intervals of a set of permutations* Π *is a pair* (R, L) *of vectors of size* n *such that:*

1. $R[i] \geq i$ *and* $L[j] \leq j$ *for all* $i, j \in \{1, 2, \ldots, n\}$,
2. $(i..j)$ *is a common interval of* Π *if and only if* $(i..j) = (i..R[i]) \cap (L[j]..j)$, *or, equivalently* $L[j] \leq i \leq j \leq R[i]$.

There are many possible generators, here is one:

Definition 30. *Let* $\sigma = (\sigma_1, \ldots, \sigma_n)$ *be a permutation of size* n. *For each element* σ_i, *we define two intervals containing* σ_i*:*

1. $IMax[\sigma_i]$ *is the largest* set *of elements larger than* σ_i *that forms an interval around* σ_i *in* σ.
2. $IMin[\sigma_i]$ *is the largest* set *of elements smaller than* σ_i *that forms an interval around* σ_i *in* σ.

And we define the following two integer vectors:

1. $Sup[\sigma_i]$ *is the largest integer such that* $(\sigma_i..Sup[\sigma_i]) \subseteq IMax[\sigma_i]$;
2. $Inf[\sigma_i]$ *is the smallest integer such that* $(Inf[\sigma_i]..\sigma_i) \subseteq IMin[\sigma_i]$

The pair of vectors (Sup, Inf) *is a generator for the common intervals of a permutation* σ.

Example 20. *IMax and IMin, Sup, and Inf of permutation* $\pi^1 = (2\ 1\ 5\ 3\ 4\ 8\ 6\ 7)$ *are:*

IMax	*Sup*	*IMin*	*Inf*
$\pi^1[1,8] = (1,2,3,4,5,6,7,8)$	8	$\pi^1[2,2] = (1)$	1
$\pi^1[1,1] = (2)$	2	$\pi^1[1,2] = (1,2)$	1
$\pi^1[3,8] = (3,4,5,6,7,8)$	8	$\pi^1[4,4] = (3)$	3
$\pi^1[5,8] = (4,6,7,8)$	4	$\pi^1[4,5] = (3,4)$	3
$\pi^1[3,3] = (5)$	5	$\pi^1[1,5] = (1,2,3,4,5)$	1
$\pi^1[6,8] = (6,7,8)$	8	$\pi^1[7,7] = (6)$	6
$\pi^1[8,8] = (7)$	7	$\pi^1[7,8] = (6,7)$	6
$\pi^1[6,6] = (8)$	8	$\pi^1[1,8] = (1,2,3,4,5,6,7,8)$	1

R and L define the following intervals:

intervals of R

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
1	–	–	–	–	–	–	–	–
2		–						
3			–	–	–	–	–	–
4				–				
5					–			
6						–	–	–
7							–	
8								–

intervals of L

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
1	–							
2	–	–						
3			–					
4			–	–				
5	–	–	–	–	–			
6						–		
7						–	–	
8	–	–	–	–	–	–	–	–

Lemma 6. *Let (R_1, L_1) and (R_2, L_2) be generators of common intervals of two sets $\Pi^1 =$ and Π^2 of permutations. The pair $(\min(R_1, R_2), \max(L1, L2))$ is a generator for the common intervals of $\Pi^1 \cup \Pi^2$.*

Proof. Interval $(i..j)$ is a common interval of $\Pi_1 \cup \Pi_2$ if and only if it is a common interval of both Π_1 and Π_2, which is equivalent to $L_1[j] \leq i \leq j \leq R_1[i]$ and $L_2[j] \leq i \leq j \leq R_2[i]$ and finally to $\max(L_1[j], L_2[j]) \leq i \leq j \leq \min(R_1[i], R_2[i])$. □

Example 21. *(R_1, L_1) and (R_2, L_2) are generators of permutations $\pi^1 = (2\ 1\ 5\ 3\ 4\ 8\ 6\ 7)$ and $\pi^2 = (1\ 3\ 2\ 4\ 5\ 7\ 6\ 8)$ are*

R_1	L_1	R_2	L_2	$R = \min(R_1, R_2)$	$L = \max(L_1, L_2)$
8	1	8	1	8	1
2	1	6	2	2	2
8	3	6	3	3	3
4	3	6	3	4	3
5	1	6	3	5	1
8	6	6	3	8	6
7	6	7	3	7	6
8	1	8	1	8	1

Given *IMin*, *Inf* can be computed in linear time with this simple algorithm:

Algorithm 4 Construction of *Inf* from *IMin*

```
1: Inf[k] ← k for k = 1..n
2: for k from 2 to n do
3:    while Inf[k] − 1 is in IMin[k] do
4:       Inf[k] ← Inf[Inf[k] − 1]
5:    end while
6: end for
```

(A similar algorithm can be designed for the computation of *Sup* from *IMax*)

The set of common intervals of k permutations of size n can be enumerated in $O(kn+n^2)$ time by computing generators (Sup, Inf) of each permutation independently,

constructing their union (R, L) as shown above, and then testing each combination of the intervals defined by generator.

The enumeration can be improved to optimal time, i.e., $O(kn + z)$, where z denotes the number common intervals defined by the generator. The improvement relies on the $Support_R$ vector: $Support_R$ a the vector that specifies for each interval $(i..R[i])$ the smallest interval $(i'..R[i'])$, in which $(i..R[i])$ is contained:

Definition 31. *The* support *of vector R is a vector $Support_R$ that refers at each position $Support_R[i]$ to the index i' of the smallest interval $(i'..R[i'])$ that is a super interval of $(i..R[i])$ and is undefined if no such interval exists.*

Example 21 (continued)**.** *$Support_R$:*

	intervals of R								$Support_R$
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	
1	–	–	–	–	–	–	–	–	/
2		–							*1*
3			–						*1*
4				–					*1*
5					–				*1*
6						–	–	–	*1*
7							–		*6*
8								–	*6*

We make the following observation:

- The support for an interval of R is only undefined for its first interval, which is always $(1..n)$.
- The support for interval $(i..R[i])$ is must be the interval corresponding to the *highest index* $i' < i$ s.t. $(i..R[i]) \subset (i'..R[i'])$.

The vector $Support_R$ can be computed in linear time from vector R if the set of intervals $\{(i..R[i]) \mid i = 1..n\}$ commutes:

Definition 32. *Two intervals A and B* commute *if $A \subseteq B$ or $B \subseteq A$ or A and B are disjoint. A set of intervals I commutes if each pair of intervals $A, B \in I$ commutes.*

Because $Support_R$ specifies for each interval defined by R interval $(i..R[i])$ the smallest interval $(i'..R[i'])$ in which $(i..R[i])$ is contained, i' must be strictly smaller than i. Thus, by maintaining a stack of previous intervals, $Support_R$ can be constructed in linear time. By definition, all intervals $(2..R[2]), ..(n..R[n])$ must be a subinterval of $(1..R[n]) = (1..n)$. The algorithm thus iterates through all intervals from $(2..R[2])to(n..R[n])$, popping intervals off the stack that are not super-intervals of the current interval as long as a proper super-interval is found.

The popped intervals cannot be super-intervals of subsequent intervals, if all intervals of R commute (i.e., one is included in the other or they are disjoint). This holds true for Sup (and consequently for the union of Sup vectors), as noted in the previous lecture.

Algorithm 5 Construction of $Support_R$ from vector R

```
1: Initialize empty stack S
   // in the following, s denotes the top of S
2: Push 1 on S
3: For i from 2 to n
4: while R[s] < i do
5:    Pop the top of S
6: end while
7: Support_R[i] ← s
8: Push i on S
```

At last, note that it is sufficient to test only whether the left bound i of the current interval $(i..R[i])$ is smaller than the right bound $R[s]$ of any previously observed interval s (which will be an interval of the stack) if one wants to know if $(i..R[i])$ is included in $(s..R[s])$. That is because (a) $i > s$ holds true by definition and (b) all intervals of R commute by supposition. The vector L and $Support_R$ are sufficient for enumerating all common intervals, as demonstrated by Algorithm 6 below.

Algorithm 6 Enumerator for Common Intervals

```
1: for j from n to 1 do
2:    i ← j
3:    while i ≥ L[j] do
4:       Report (i..j) // interval (i..j) is a common interval
5:       i ← Support_R[i]
6:    end while
7: end for
```

The algorithm simply iterates through all possible right bounds j of a common interval in decreasing order. Recall that, if $(i..j)$ is a common interval, it must hold that $L[j] \leq i \leq j \leq R[i]$.

Let $(s..R[s]) = (Support_R[i], R[Support_R[i]])$ be the smallest interval that contains $(i..R[i])$. Recall that each interval of R is maximal, it is the largest interval around element i that contains only elements larger i, with $R[i]$ being the largest one. We have: $(i..R[i]) \subset (s..R[s])$ and thus $R[i] \leq R[s]$. Observe that due to maximality, $R[i] = R[s]$. The idea of the algorithm is to continuously decrease $j = R[i]$ in the outer loop, thereby allowing i to take on values corresponding to the left bound of the supporting interval, i.e. $s = Support_R[i]$, in the inner loop. Interval $(s..R[i])$ is the smallest super-interval of $(i..R[i])$ by definition of $Support_R$. It is also the smallest possible common interval that is larger than $(i..R[i])$ for fixed right bound $R[i]$. It then suffices to execute the test for common intervals for the left bound, i.e, $L[j] \leq i$. As long as this condition is true, the algorithm will report common interval $(i..R[i])$ and decrease the left bound i to that of its supporting interval.

The running time of the algorithm is $O(kn + z)$, where z is the number of common

intervals of (R, L). The time complexity is apparent: In each step the algorithm either finds a common interval, or continues to the next right bound $j \leftarrow j - 1$.