Common intervals
MPRI 2015–2016

Michel Habib
habib@liafa.univ-Paris-Diderot.fr
http://www.liafa.univ-Paris-Diderot.fr/~habib

Sophie Germain, octobre 2015
Schedule

Remarks on partitive families

Applications
  Common connected sets
  Maximal common intervals

Common Intervals

A more precise description of the problem

Finding all common intervals?

Known results

The geometric approach by Ismael Belghiti

The main algorithm
  Finding One Non-Trivial Common Interval
  Building the tree
  Listing all and extensions
Partitive Families

Lemma

If \( M \) and \( M' \) are two overlapping modules then

1. \( M \setminus M' \) is a module
2. \( M \cap M' \) is a module
3. \( M \cup M' \) is a module
4. \( M \Delta M' \) is a module

A family satisfying (i) - (iv) is called a **partitive family**
A family satisfying (i) - (iii)) is called a **weak partitive family**
Remarks on the module properties

- (ii) is always true, even if $M$ and $M'$ are not overlapping.
- (iii) could be written that way:
  If $M \cap M' \neq \emptyset$ then $M \cup M'$ is a module.
- (iv) is not a consequence of (i) and (iii) since $M \setminus M'$ and $M' \setminus M$ do not overlap. And overlapping is really needed to prove (i) for modules.
Partitive Families

We call Partitive Family a set family $\mathcal{F}$ on a ground set $X$ such that:

1. $\emptyset \in \mathcal{F}$, $\forall x \in X$, $\{x\} \in \mathcal{F}$ and $X \in \mathcal{F}$
2. $\forall A, B \in \mathcal{F}$, $A \cap B \in \mathcal{F}$
3. $\forall A, B \in \mathcal{F}$ if $A \cap B \neq \emptyset$ then $A \cup B \in \mathcal{F}$
4. If $A \in \mathcal{F}$ and $B \in \mathcal{F}$ overlap then $A \setminus B \in \mathcal{F}$
5. If $A \in \mathcal{F}$ and $B \in \mathcal{F}$ overlap then $A \Delta B \in \mathcal{F}$

A family satisfying (i) - (iv) is called a partitive family.
A family satisfying (i) - (iii) is called a weak partitive family.
Common connected sets

Definition

Input: Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two undirected graphs on the same vertex set $V$.

Results: the sets $A \subseteq V$ maximal under inclusion such that $G_1(A)$ and $G_2(A)$ are both connected.
Denoted by maximal common connected sets.

Origins

Given 2 genomic sequences $\tau_1, \tau_2$, in which genes have been identified.
$V$ is the set of genes. Put an edge between two genes in $E_i$ if the distance between the two genes in the sequence $\tau_i$ is less than $\delta$. 
More on the application

The input graphs could be considered as interval graphs.

The problem applies for a series of $k$ genomic sequences.

Need for an efficient algorithm.

First remarks

FAQ
These maximal connected sets are not necessarily connected components of one of the graphs

Common edges
If $ab \in E_1 \cap E_2$, one can contract the vertices $a$ and $b$ to a single vertex and goes on recursively if needed. Filtering in $O(n + m)$ yields an instance for which $E_1 \cap E_2 = \emptyset$.

Wlog
Let us assume $E_1 \cap E_2 = \emptyset$. 
Naive algorithm

Lemma
If for some $A \subset V$, there is no edge between $A$ and $V - A$ in one of the graphs $G_i$, then the maximal common connected sets are those of $G_1(A)$ and $G_2(A)$ plus those of $G_1(V - A)$ and $G_2(V - A)$.

Hard case
$G_1$ is supposed to be connected, and $G_2$ is not connected.

Naive algorithm
A naive algorithm which recursively computes connected components and refines the current partition of vertices requires $O(n(n + m))$, where $n = |V|$ and $m = |E_1| + |E_2|$, and this bound is tight.
Relationships with modular decomposition

Observation

For a maximal common set $A$, either $|A| = 1$, or $|A| \geq 4$. As for example a $P_4$ and its complement.

Particular case

When $G_2 = \overline{G_1}$, the common connected sets are exactly the maximal prime nodes in the modular decomposition tree of $G_1$. Can be computed in linear time using modular decomposition algorithms.
Maximal common intervals

Maximal common interval to a set of permutations.
T. Uno and M. Yagura, linear time algorithm, Algorithmica 2000

Equivalent to find maximal non trivial common connected set when $G_1$ and $G_2$ are paths.

Equivalent to compute the modular decomposition of a permutation graph, when a representation is provided.
Several new $O(n)$ algorithms:
F. de Montgolfier, R. Mc Connell 2004
A. Bergeron, F. de Montgolfier, M. Raffinot, ESA 2005
B.M. Bui Xuan, mh, C. Paul, ISAAC 2005. . . .
Even when $G_1$ and $G_2$ are forests it is not easy to design a linear algorithm.
A nasty recursive example:

\( \mathcal{G}_2 = (G_1^2, G_2^2) : \)

\( \mathcal{G}_q = (G_1^q, G_2^q) : \)

\( \mathcal{G}_4 = (G_1^4, G_2^4) : \)
Main trick

Avoid the biggest component

A data structure that maintains an entry for each component
Ensures a logarithmic behaviour.
Bui Xuan, M.H., Paul, TCS 2008.
## Applications

### Maximal common intervals

<table>
<thead>
<tr>
<th>Graph classes</th>
<th>Generic approach</th>
<th>Conjecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>forests of trees</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>interval graphs</td>
<td>$O(n + m \log n)$</td>
<td>$O(n + m)$</td>
</tr>
<tr>
<td>planar graphs</td>
<td>$O(n \log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>permutation graphs</td>
<td>$O(n + m \log^2 n)$</td>
<td>$O(n + m)$</td>
</tr>
<tr>
<td>arbitrary graphs</td>
<td>$O(n + m \log^2 n)$</td>
<td>$O(n + m)$</td>
</tr>
</tbody>
</table>

**Figure:** Common connected component computation time, with $n$ the number of vertices, $m$ the total number of edges.
For arbitrary graphs, we need to use a sophisticated data structure called ET-tree from Holm, de Lichtenberg and Thorup ACM 98 introduced for fully-dynamic algorithms for connectivity.

A nice algorithmic problem:
Find a linear time algorithm that computes "common intervals" of a permutation and a tree. (i.e. an interval and a subtree).
Let $\tau$ and $\sigma$ be two permutations on $[1, n]$. Without loss of generality (wlog) $\tau$ is supposed to be the identity.

2 permutations

$\tau = [1, 2, 3, 4, 5, 6, 7, 8]$

$\sigma = [4, 6, 3, 5, 8, 2, 7, 1]$

Definition

Two intervals $I, J \subseteq [1, n]$ are common intervals of $\tau$ and $\sigma$ if $\tau(I) = \sigma(J)$ as subsets of $[1, n]$. 

Example

$\tau = 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8$

$\sigma = 4 \ 6 \ 3 \ 5 \ 8 \ 2 \ 7 \ 1$

The ordering of the elements may differ.

$[3, 6] \ [1, 4]$ are the unique non trivial maximal common intervals of $\tau$ and $\sigma$.

$\tau([3, 6]) = \sigma([1, 4]) = \{3, 4, 5, 6\}$
Other presentation: Common Connected Sets

Given two chains on $X$: find the subsets of $X$ connected in both chains.

**Figure**: Common Connected Sets of Two Chains

This is equivalent to the common intervals of: $(2, 6, 1, 3, 5, 4)$ and $(6, 1, 2, 4, 3, 5)$. 
Problems

The size of the data is in $O(n)$.

1. Propose an algorithm which computes a non trivial maximal common interval

2. Propose an algorithm which computes all non trivial common intervals

3. Same problems for $k$ permutations

4. Same problems with some fixed number of errors still widely open
Intervals of a Permutation

$I \subseteq [1, n]$ is an **interval** of a permutation $P$ if

- there exists $x, y$ such that $P([x, y]) = I$
- i.e., the elements of $I$ appear consecutively in $P$.

Example: $\{1, 2, 6\} = P([1, 3])$ and $\{1, 3, 5, 6\} = P([3, 6])$ are intervals of $P = (2, 6, 1, 3, 5, 4)$
Common Intervals of Permutations

Given a set of permutations \( \mathcal{P} = \{P_1, \ldots, P_k\} \) of \([1, n]\), \( I \) is a common interval when \( I \) is an interval of each permutation of \( \mathcal{P} \).

Examples: \( \{1, 2, 6\} \) is a common interval of: \((2, 6, 1, 3, 5, 4)\), \((6, 1, 2, 4, 3, 5)\) and \((3, 4, 1, 6, 2, 5)\).
The problem

Our problem is:

- Given permutations $P_1, \ldots, P_k$,
- "find all their common intervals"
Case of two permutations

We will restrict the problem to the case of two permutations $P_1$ and $P_2$.

Transformation of the input:

- Renumber the elements such that $P_1 = Id_n$.
- i.e., just consider $P = P_2 \circ P_1^{-1}$

In $P$, we consider the $[x, y]$ such that $P([x, y])$ is an integer interval.
Example and representation

\[(2, 6, 1, 3, 5, 4)\]
\[(6, 1, 2, 4, 3, 5)\]
\[(1, 2, 3, 4, 5, 6)\]
\[(2, 3, 1, 6, 4, 5)\]
What does ”finding all common intervals” mean?

One can:

1. Listing all common intervals using an algorithm output sensitive. Which is possible since there are at most $n^2$ common intervals.

2. Compute a structure which represents all common intervals.
Listing all the common intervals

   They proposed an $O(n + K)$ algorithm to enumerate all common intervals of two permutations ($K$ is the number of common intervals).

   A better proof based on the submodularity of splitter number of an interval.
Using Generators


They introduced the notion of generator.

$(R, L)$ is a generator when:

- $\forall i \in [1, n], R[i] \geq i$
- $\forall i \in [1, n], L[i] \leq i$
- $[x, y]$ is a common interval $\iff x \geq L[y]$ and $y \leq R[x]$. 


Using Generators

Such a couple can be constructed as follow:

1. $R(x) = \max\{y | [x, y] \text{ is a common interval} \}$
2. $L(y) = \min\{x | [x, y] \text{ is a common interval} \}$

A simple data structure that represents the family!
Generators

- size 2 arrays of size $n$
- $O(1)$ to check if an interval is a common interval.
- $O(n^2)$ to list all common intervals. (Could be improved to $O(n + K)$?)
Example and representation
Example and representation
Example and representation
What common intervals mean in this matricial representation?

A common interval is just a "complete square of points", where complete means exactly one pixel per line and per column. $[x, y]$ such that $P([x, y])$ is an integer interval means exactly a complete square.
The common intervals of $P$ form a **weakly partitive family**. If two common intervals $A, B$ overlaps then:

- $A \cup B$ is a common interval
- $A \cap B$ is a common interval
- $A \setminus B$ is a common interval
Weakly Partitive Family
Weakly Partitive Family
Weakly Partitive Family can be easily represented!
Representing the tree of overlap-free members

Note: A set is said to be overlap-free, if it does not overlap another one member of the family.
Three Quotients

A quotient is the permutation obtained by contracting the sons of a node in the tree of overlap-free.

Three Kinds:
- Prime: no non-trivial common interval
- Increasing
- Decreasing
Three Quotients
Tree Decomposition

**Figure:** Tree decomposition of the permutation
(6, 7, 8, 9, 3, 5, 1, 4, 2, 14, 16, 15, 17, 18, 12, 10, 13, 11)
First Challenge

Find one non-trivial common interval.

Main idea

The algorithm uses only one main scan and finds the common intervals sequentially.
The main algorithm
Finding One Non-Trivial Common Interval

Anim

[animAlgo]
Notion of potential beginning

**Definition**: The index $x$ is a **potential beginning** for $y \geq x$ when:

- $\nexists \; z_1 < x, x < z_2 \leq y, P(x) < P(z_1) < P(z_2)$
- $\nexists \; z_1 < x, x < z_2 \leq y, P(x) > P(z_1) > P(z_2)$
- In both cases, $x$ is the central point of a $V$. 
Anim

\[ V \text{ can be incrementally computed during the scan, when considering a new vertex.} \]
**lemma**: If \( x < x' \) are potential beginnings for \( y \) then:

\[ x \text{ potential beginning of } y + 1 \Rightarrow x' \text{ potential beginning of } y + 1 \]

**proof**

If we suppose that a contrario to \( x, x' \) belongs to a \( V \) with \( y \). There exists \( z \) (let us take the first case) such that:

\[ z < x' < x < y \text{ such that } P(x') < P(z_1) < P(y). \]

Either \( P(x') < P(z) \) and then \( x' \) has a \( V = (z, x', y) \) with \( y \) and cannot be a potential beginner.

Else \( P(z) < P(x') \) and then \( x \) has a \( V = (z, x, x') \) with \( x' \) which would have been detected when considering \( x' \).
Stack of potential beginnings

Then the set potential beginnings behave as a stack!
Stack of potential beginnings

**Lemma**: The beginnings of the common intervals that ends at \( y \) form a suffix of the stack.

We only have to check if the top of the stack is the left corner of a common interval which as for right vertex the current vertex. In the YES case we delete these two vertices and extract the corresponding common interval. In the NO case, the current vertex is considered as a potential beginner (since he has no \( V \)).
Only the potential beginners have to be stored, the others vertices can be ignored
Maintaining the stack

We will define:

- \( \text{miniAmongGreaterOnLeft}(x) = \min\{P(x') | x' < x, P(x') > P(x)\} \)
- \( \text{maxiAmongSmallerOnLeft}(x) = \max\{P(x') | x' < x, P(x') < P(x)\} \)

These values can be precomputed easily in linear time as a preprocessing.
The main algorithm

Finding One Non-Trivial Common Interval
Maintaining the stack

When we add $y$, we pop the top $x$:

- if $\text{miniAmongGreaterOnLeft}(x) < P(y)$
- or $\text{maxiAmongSmallerOnLeft}(x) > P(y)$

% This test whether the introduction of $y$ creates a $V$ for $x$. If YES $x$ cannot be anymore a potential beginning.
The Algorithm

The core of the algorithm is:

\[
\text{PotBegin} \leftarrow \{1\}
\]

For \( y \) from 2 to \( n \):

While \( \min\text{AmongGreaterOnLeft}(\text{PotBegin}.\text{top()}) < P(y) \)

or

\( \max\text{AmongSmallerOnLeft}(\text{PotBegin}.\text{top()}) > P(y) \):

\text{PotBegin}.\text{pop()}

Test if \([\text{PotBegin}.\text{top()}, y]\) forms a common interval

\text{PotBegin}.\text{Push}(y)
Common intervals  MPRI 2015–2016
The main algorithm
Finding One Non-Trivial Common Interval

Check the common intervals

How to test if $[x, y]$ is a common interval?

We can check that $\maxi(x, y) - \mini(x, y) = y - x$ where:
% test of the complete square.

- $\maxi(x, y) = \max\{P(z) | z \in [x, y]\}$
- $\mini(x, y) = \min\{P(z) | z \in [x, y]\}$

We can easily do this check by maintaining some maximums and minimums.
Main invariants

- All common intervals so far have been detected
- The stack contains all remaining potential beginnings (with no $V$) ordered increasingly.
How to build the tree?

Roughly speaking:

Proceed as before contracting the common intervals on the fly.
The main algorithm

Building the tree

[anim de l’algo]
Three cases

We have three cases:

- We extend an increasing node
- We extend a decreasing one
- We create a prime super-node
The Algorithm

When we add $y$, do as long as possible:

- An extension (increasing or decreasing)
- The creation of a prime

With priority to the extensions
The Algorithm

For y from 1 to n :
    CommonintervalStack.Push(Commoninterval({y}))
    stable <- False
While not stable :
    stable <- True
    If tryExtension() or tryPrimeCreation() 
       stable <- false

(We use lazy "or")
Conclusion and Generalizations

- We obtain a very simple $O(n)$ algorithm for this problem.
- Very easy to adapt the algorithm to list all common intervals
- We extended this result to the cases of circular permutations.
- For three permutations we play with a three dimensional grid.
Another explanation for weakly partitive family

- A permutation graph is the intersection graph of segments between 2 parallel lines.
- There is a bijection between pairs of permutations and permutation graphs.
- A common interval of $P_1$ and $P_2$ implies $I$ is a module of $G(P_1, P_2)$.
- The converse can be false because of series and parallel nodes (complete or empty subgraphs).
  Any subset is a module is transformed into any interval is a common interval.
\textbf{Figure:} \{3, 4\} is a module of $G$ and a common interval of $P_1$ and $P_2$
Example

\[ \tau = \begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\end{array} \]

\[ \sigma = \begin{array}{cccccccc}
4 & 6 & 3 & 5 & 8 & 2 & 7 & 1 \\
\end{array} \]

The ordering of the elements may differ.

[3, 6] and [1, 4] are the unique non trivial maximal common intervals of \( \tau \) and \( \sigma \).

\[ \tau([3, 6]) = \sigma([1, 4]) = \{3, 4, 5, 6\} \]

\{3, 4, 5, 6\} is the only non trivial module of \( G(\tau, \sigma) \).
Partitive Families

We call Partitive Family a set family $F$ on a ground set $X$ such that:

- (0) $\emptyset \in F$, $\forall x \in X$, $\{x\} \in F$ and $X \in F$
- (i) $\forall A, B \in F$, $A \cap B \in F$
- (ii) $\forall A, B \in F$ if $A \cap B \neq \emptyset$ then $A \cup B \in F$
- (iii) If $A \in F$ and $B \in F$ overlap then $A \setminus B \in F$
- (iv) If $A \in F$ and $B \in F$ overlap then $A \Delta B \in F$

- A family satisfying (i) - (iv) is called a **partitive family**
- A family satisfying (i) - (iii) is called a **weak partitive family**
When you find a partitive family, you know that there exists a tree representation of the family with 2 types of nodes:

- Primes (undecomposable ones) and Fragile (i.e., highly decomposable ones)
- It suffices to understand which are the fragile nodes: how many kinds...
So far we have considered at least three distinct representations of the problem, and it was helpful!

1. common intervals of permutations
2. common connected components of graphs
3. Modules of a permutation graph
Some references