Solving C1P components

- Connected components of Strictly Overlap Graph
- Process each row in turn
- Place each row according to a previously placed neighbor
- To fix direction, need to have already processed neighbor of neighbor
- Process rows in DFS

Algorithm *Place*

- **input**: $w \rightarrow v \rightarrow u$ (w and v can be nil)
- **if** v = nil **and** w = nil: place 1s of u consecutively
- **if** w = nil: place $v \setminus u$, $v \cap u$, $u \setminus v$
- **if** $|u \cap w| < \min(|u \cap v|, |v \cap w|)$:
  w, $v \setminus u$, $v \cap u$, $u \setminus v$
- **else**:
  $v \setminus u$, $v \cap u$, $u \setminus v$, w
- Check consistency of columns sets

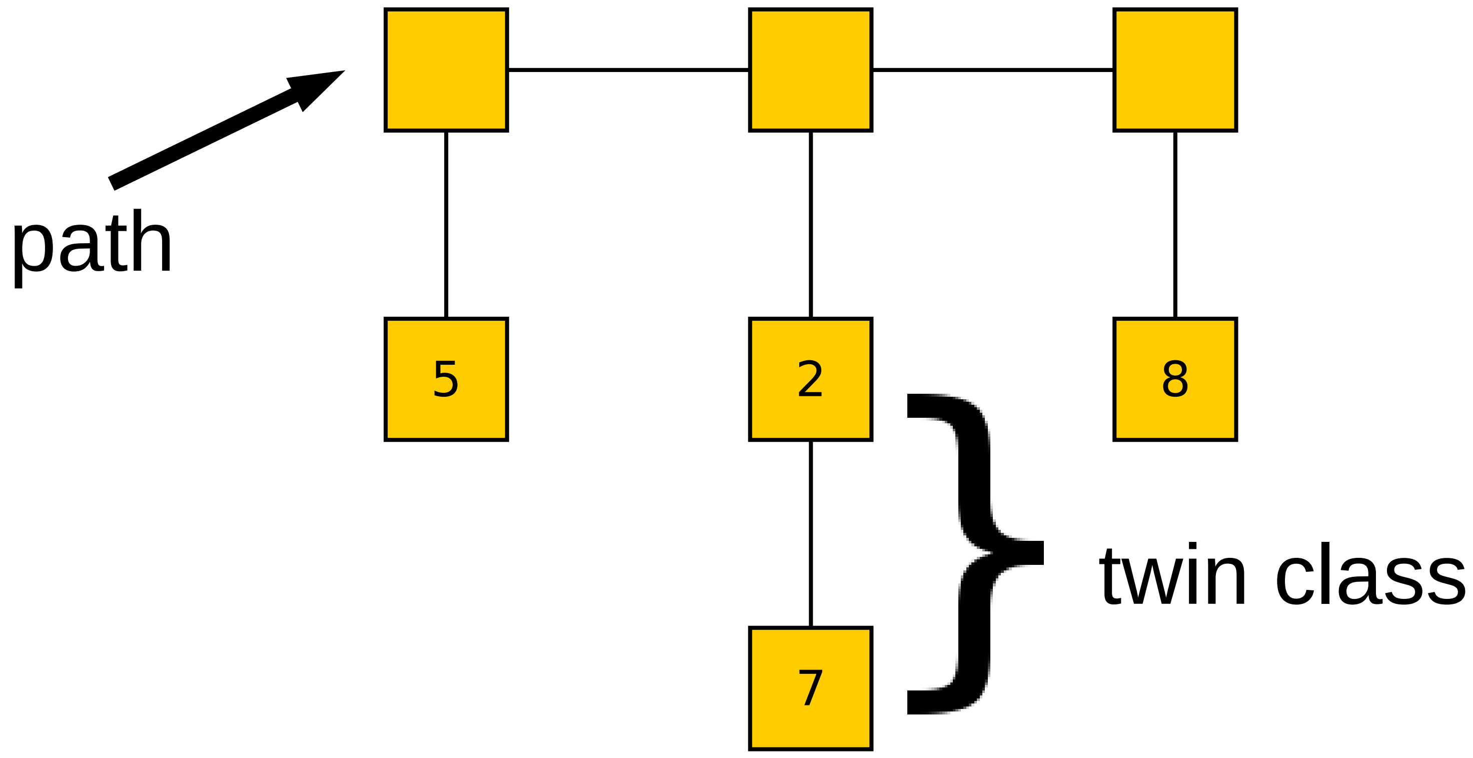
Issues

- Twin elements: belong to exactly the same sets
- Twin elements are grouped in twin classes (column sets)
- How to handle twin sets?
- Complexity: $O(mn) \rightarrow O(m + n + f)$, where

  $f = \sum|S|$

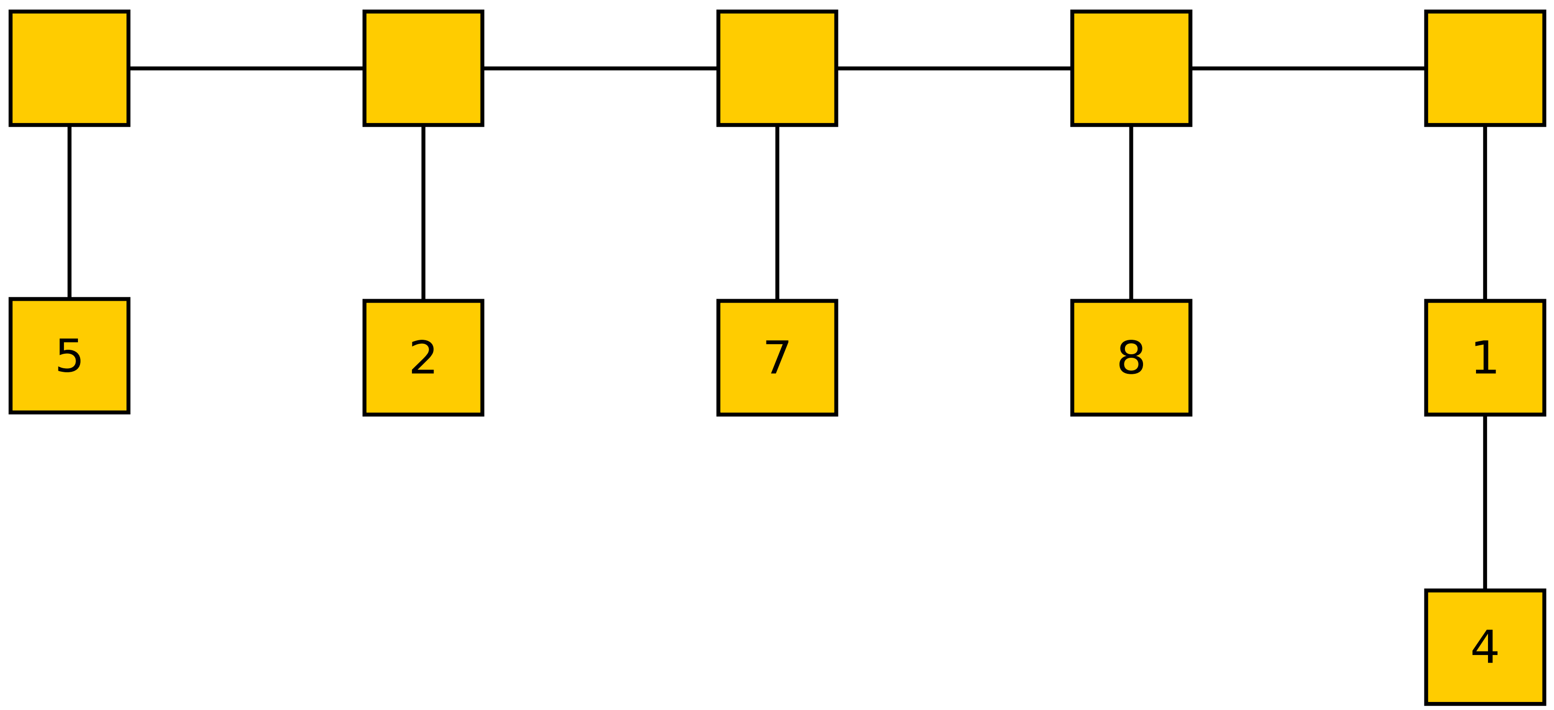
Twin classes in a Path

- After adding {2,7,8} and {2,5,7}



Twin classes in a Path

- After adding {2,7,8} , {2,5,7} , and {1,4,7,8}



Improved placement algorithm

```
• path ← [ ] ;        C1P ← **true**
• for each set S in DFS order :
      color each x $\in$ S in its class (possibly newclass)
      for each partial class C :
          cut uncolored elements from C make a new class
          place uncolored subclass near empty neighbor
          if no such neighbor : C1P ← false
      if nonconsecutive full classes : C1P ← false
      if there is a newclass :
          if both path extremities empty : C1P ← false
          place newclass in path extremity, preferably full
      remove colors
```