Networks on Cake Cutting

Ramsundar Anandanarayanan, Puranjay Datta, Anish Kulkarni

October 13, 2023

Cake Cutting

The general cake-cutting problem:

- **What we have**:
  - A *cake* , represented by interval $\mathcal{C} = [0, 1]$.
  - Player set $N = \{1, \dots n\}$ (represented by $[n]$).
  - *Valuation functions* for each player, $v_i : \mathcal{P}(\mathcal{C}) \rightarrow [0, 1] \quad \forall\, i \in N$.
  - Our assumptions for each v_i - *Additive* and *Divisible*.
  - *Additive* - $\forall\, X, Y \subseteq \mathcal{C},\; v_i(X \cup Y) + v_i(X \cap Y) = v_i(X) + v_i(Y)$
  - *Divisible* - $\forall\, X \subseteq \mathcal{C}, \lambda \in [0, 1], \exists Y \subseteq X \; s.t.\; v_i(Y) = \lambda v_i(X)$
  - Furthermore, we assume each v_i is normalized, i.e. $v_i(\mathcal{C}) = 1$.

Cake Cutting

- **What we want**:
  - An allocation $\{A_1, \ldots, A_n\}$ (a partition of $\mathcal{C}$) that, either exactly or approximately, establishes some notion of *fairness*.
  - Possible criterion for fairness - *Envy-Freeness*, *Proportionality*.
  - *Envy-Freeness* -
    $\forall\, i, j \in N,\ v_i(A_i) \geq v_i(A_j)$
  - *Proportionality* -
    $\forall\, i \in N,\ v_i(A_i) \geq 1/n$
  - We will mainly focus on *envy-freeness*.

Networks

- **The problem**: Defining a global fairness criterion is often overly restrictive and hard to obtain. Most practical scenarios involve allocating over some social or institutional network.
- The problem we studied involved imposing a network structure over N and then studying fair cake-cutting. This graphical framework was first proposed in [BQZ17].
- In networked fairness, an agent i is only concerned about *local fairness*, i.e. fairness w.r.t his neighbours $N(i)$ in the network.
- *Envy-Freeness* (in network) - $\forall j \in N(i),\ v_i(A_i) \geq v_i(A_j)$
- *Proportionality* (in network) - $v_i(A_i) \geq \Sigma_{j \in N(i)} v_i(A_j)/|N(i)|$

An important observation: An allocation that is *envy-free* over a graph G, will also be envy-free over any subgraph G' of G. (The same cannot be said *for proportional* allocations)

Motivation

Existing results

- *Discrete* and *bounded* algorithms for normal envy-free division for 4 agents[1] and n agents[2]. By our previous observation, this means we already have an algorithm for envy-freeness over any network.
- However, these algorithms require a large number of queries and cuts: the 4-agent algorithm can go upto *203 cuts*, and the general algorithm has $O(n^{n^{n^{n^{n^{n}}}}})$ queries.
- Thus, it is necessary to focus on simpler and more efficient algorithms for fairness over networks with special properties.

[1] AM16b.
[2] AM16a.

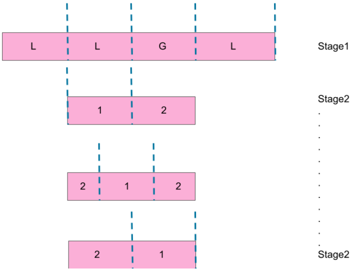
Austin Cut

- This leads us to the **Austin Cut** algorithm.
- A *continuous*, *moving-knife* procedure.
- **Given**: (i, j, m, C), where i, j are players, $m \in \mathbb{N}$, and C is some cake portion.
- AustinCut(i, j, m, C) partitions C into m parts, s.t. for every piece P, $v_i(P) = v_i(C)/m$ and $v_j(P) = v_j(C)/m$ both hold.
- Requires atmost $2m$ cuts. However, the drawback is of course, that it is continuous. It's currently unknown if this can be implemented by a discrete algorithm.
- [BQZ17] utilizes Austin Cut to give fair and efficient algorithms for *tree networks* and *descendant graph networks*.

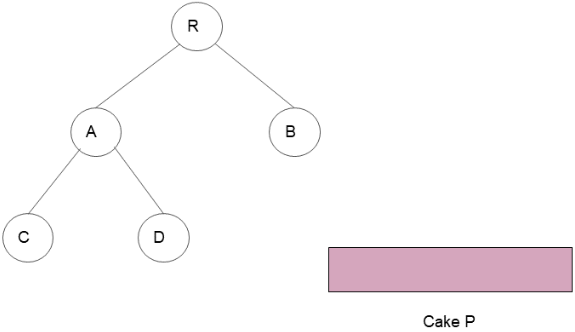
Austin Cut-2 Players, k equal pieces

WLOG, we'll assume C represents $\mathcal{C}$.

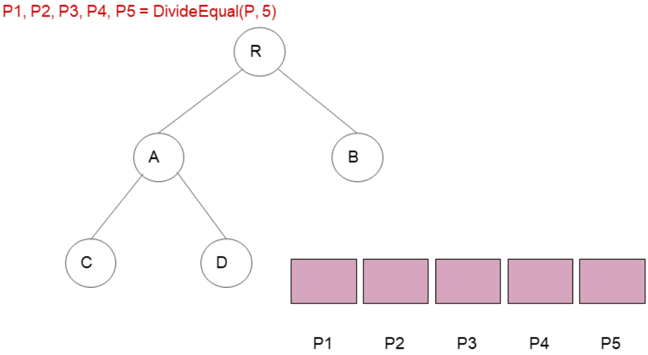
1. i marks C into m equal pieces (according to v_i).
2. **If** there is a piece A s.t. $v_j(A) = 1/k$, take A and stop.
3. **Else** there are adj. pieces A, B s.t. $v_j(A) < 1/k$ and $v_j(B) > 1/k$.
4. i can keep k knives/pointers starting at A, and move them towards B in such a way that the portion bet. them always has v_i value $= 1/m$.
5. By IVT, j will find a portion P bet. the knives at some point s.t. $v_j(P) = 1/m$ too. Take P and stop.
6. Repeat steps 2-5 until m pieces are obtained.



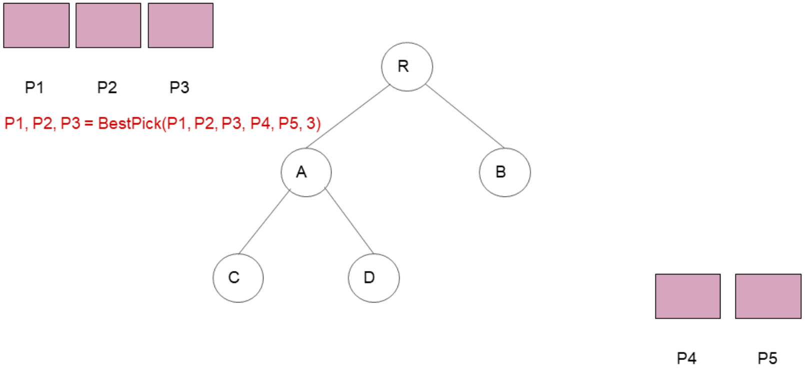
Tree algorithm [BQZ17]



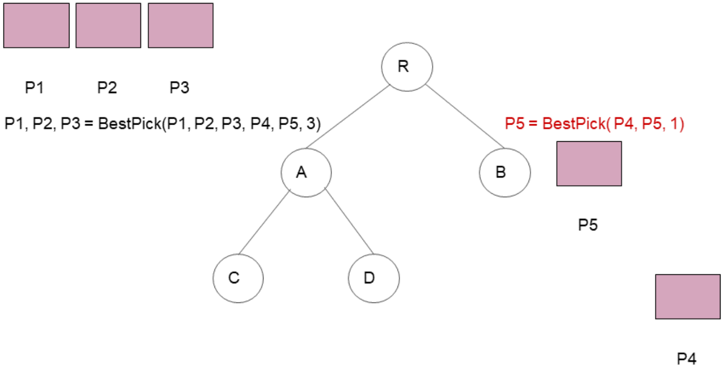
Tree algorithm [BQZ17]



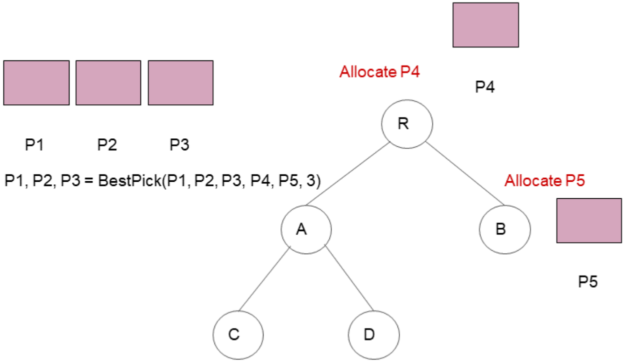
Tree algorithm [BQZ17]



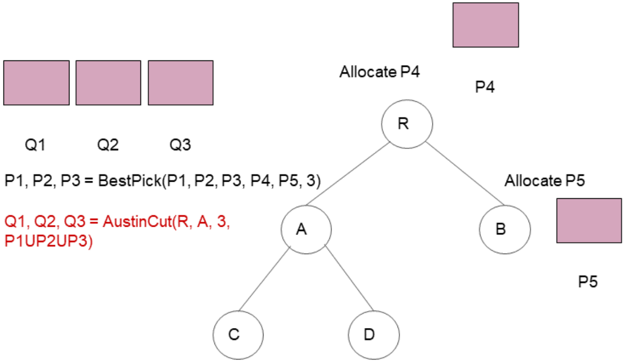
Tree algorithm [BQZ17]



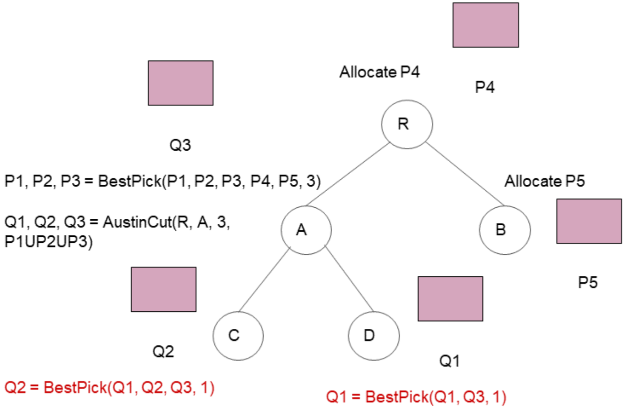
Tree algorithm [BQZ17]



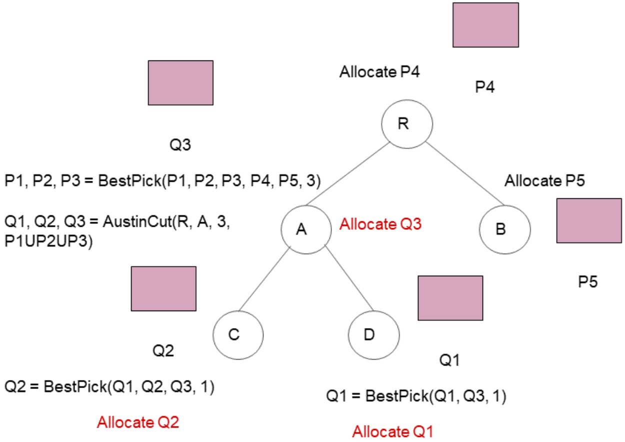
Tree algorithm [BQZ17]



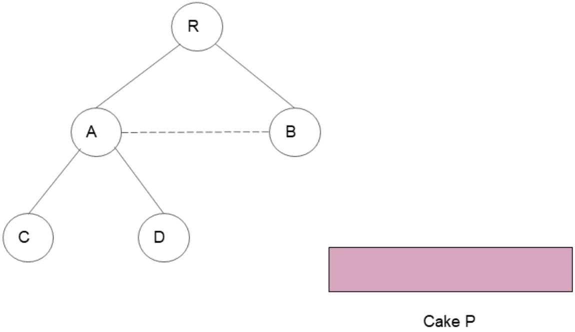
Tree algorithm [BQZ17]



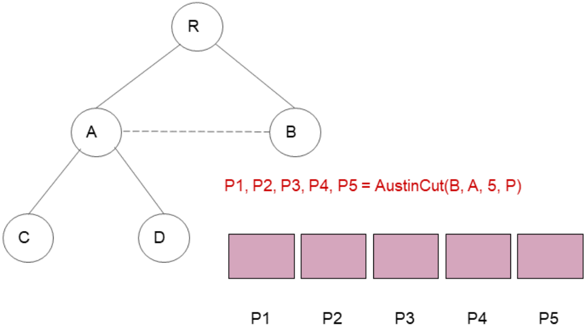
Tree algorithm [BQZ17]



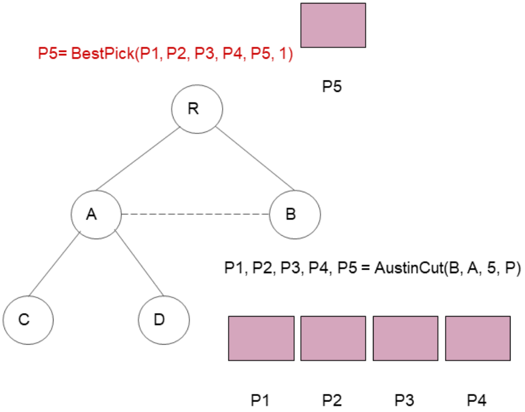
EXTENSION: Tree with an additional edge at level 1



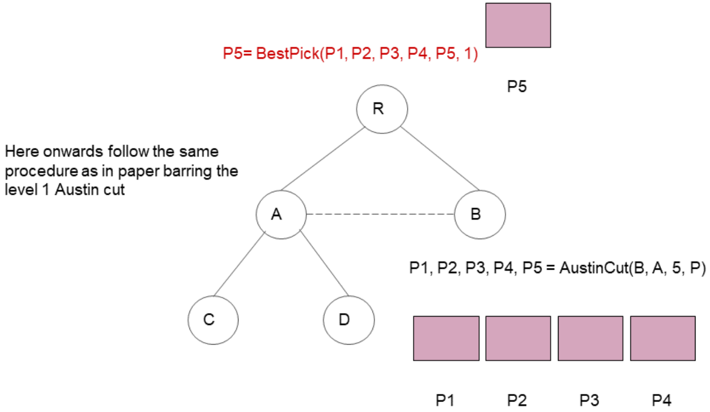
EXTENSION: Tree with an additional edge at level 1



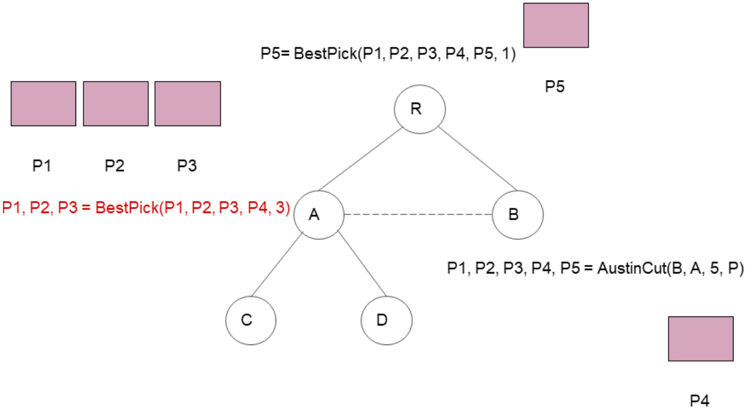
EXTENSION: Tree with an additional edge at level 1



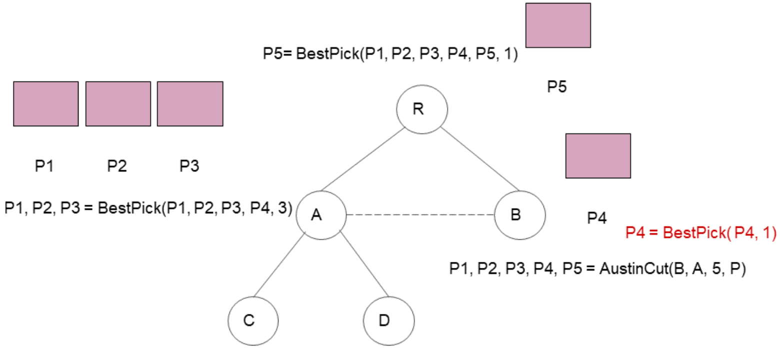
EXTENSION: Tree with an additional edge at level 1



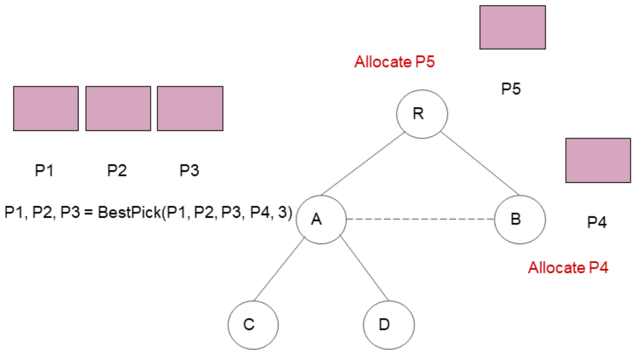
EXTENSION: Tree with an additional edge at level 1



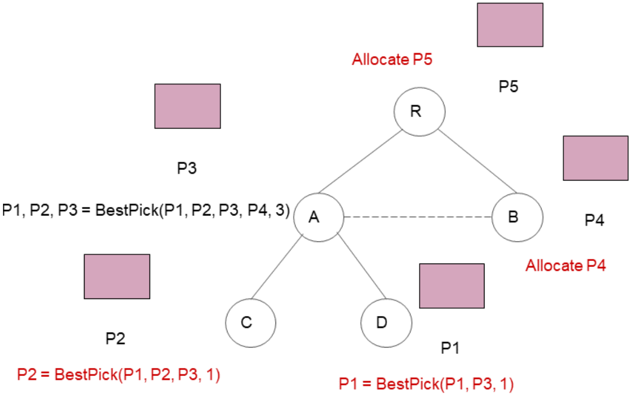
EXTENSION: Tree with an additional edge at level 1



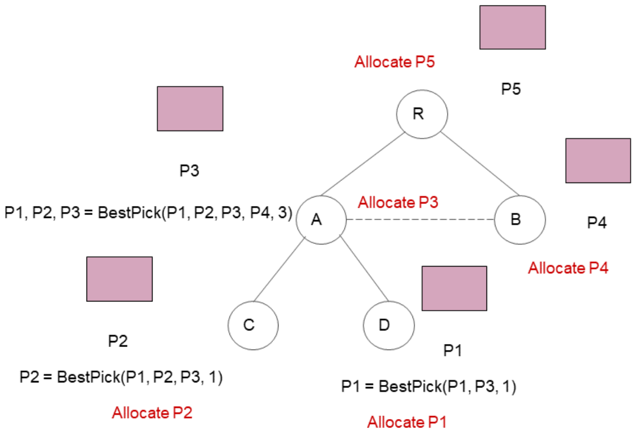
EXTENSION: Tree with an additional edge at level 1



EXTENSION: Tree with an additional edge at level 1



EXTENSION: Tree with an additional edge at level 1



Brams–Taylor–Zwicker procedure

- Global envy-free cake cutting among 4 players [BTZ97]
- Non-discrete algorithm since it uses Austin cut as a sub-procedure.
- 13 cuts are required. (Can be optimized to 11)

AustinCut$(1, 2, 4, \mathcal{C})$
Trim(3) $\triangleright$ Player 3 creates two-way tie for largest piece
ChooseOrder(4, 3, 2, 1, $\mathcal{C} \setminus \{\text{trimming}\}$) $\triangleright$ Either 3 or 4 chooses trimmed piece
if trimmed piece $\to$ 3 **then**
    AustinCut(4, 1, 4, trimming) $\triangleright$ trimming = smaller trimmed piece (w.r.t. 3)
    ChooseOrder(3, 2, 4, 1, trimming)
else
    AustinCut(3, 1, 4, trimming)
    ChooseOrder(4, 2, 3, 1, trimming)
end if

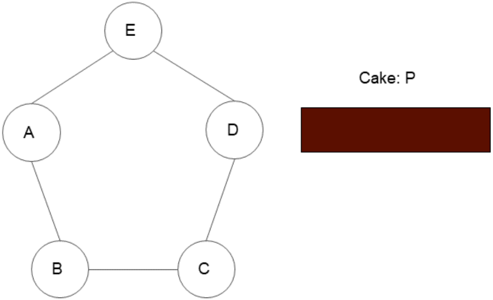
Structures

- Cycles(length: 5, 6)
- Cliques connected via a Bridge (clique with 3 vertices)
- Other Miscellaneous Structures

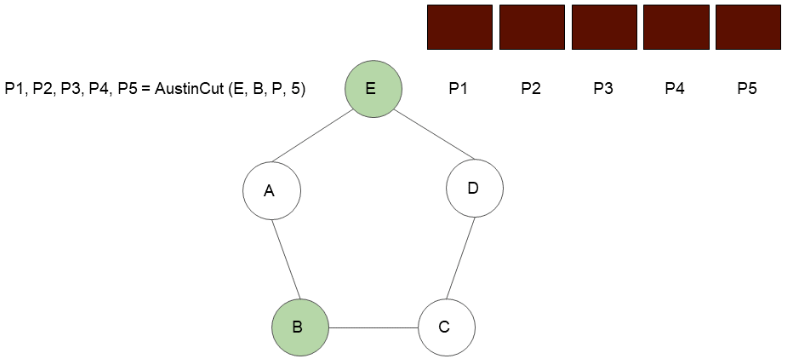
Structures

- Cycles(length: 5, 6)
- Cliques connected via a Bridge (clique with 3 vertices)
- Other Miscellaneous Structures

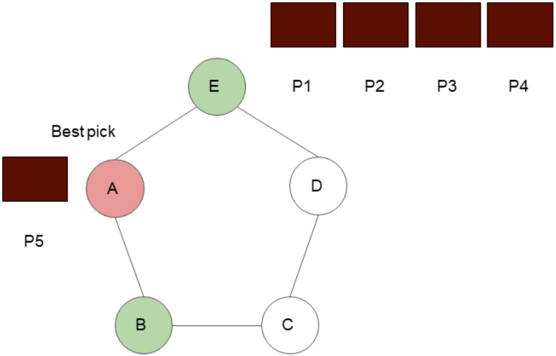
Cycle (C_5)



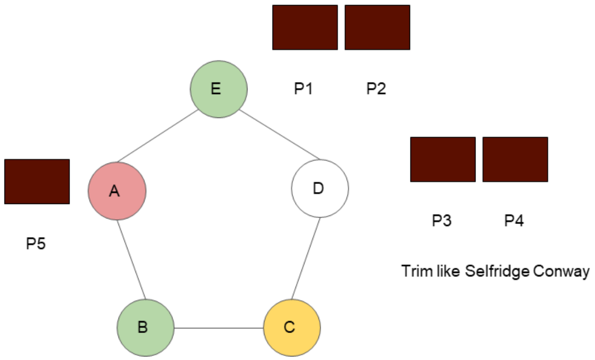
Cycle (C_5)



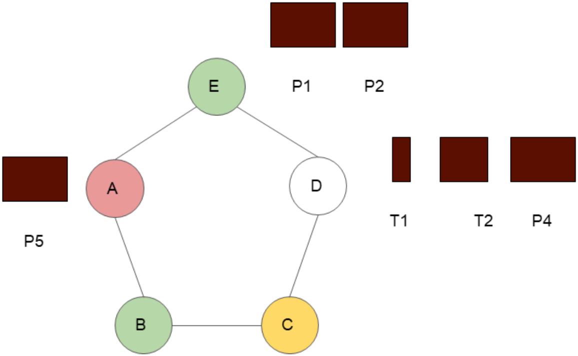
Cycle (C_5)



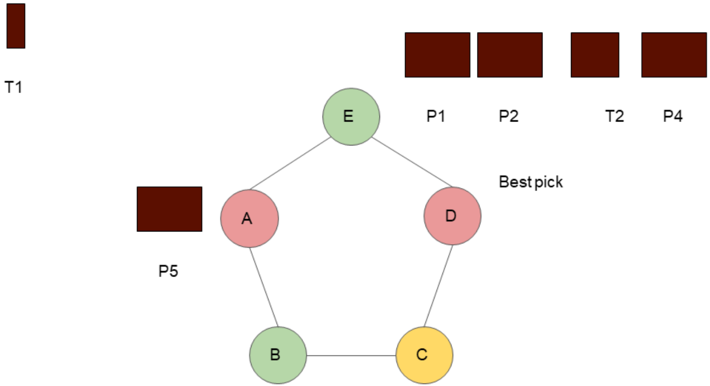
Cycle (C_5)



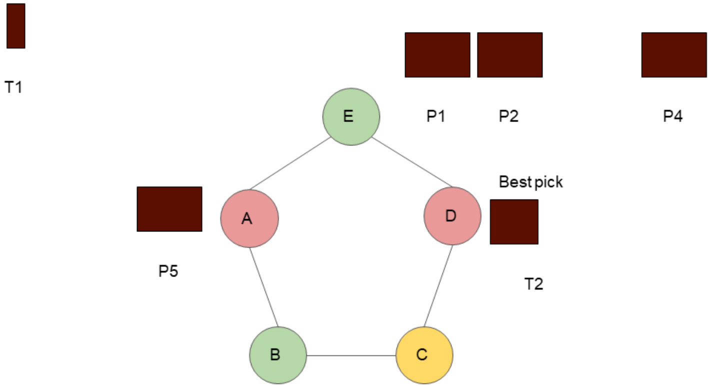
Cycle (C_5)



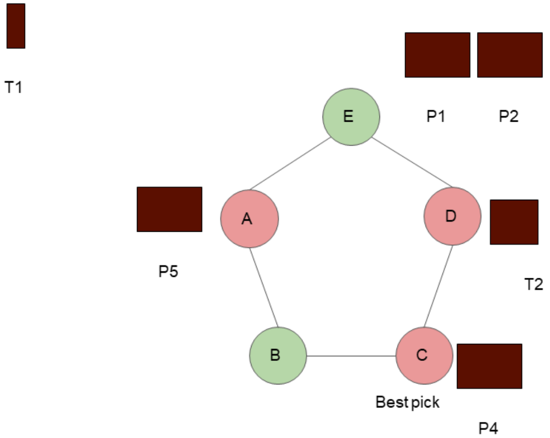
Cycle (C_5)



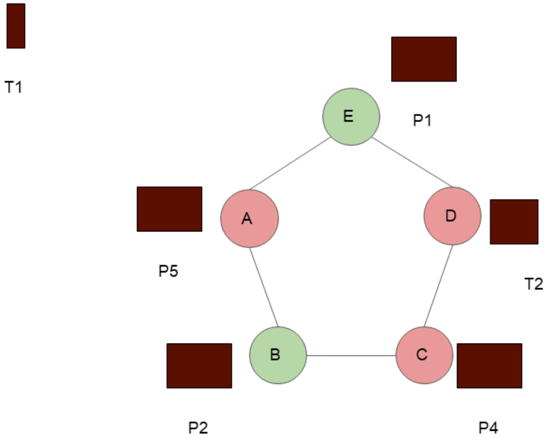
Cycle (C_5)



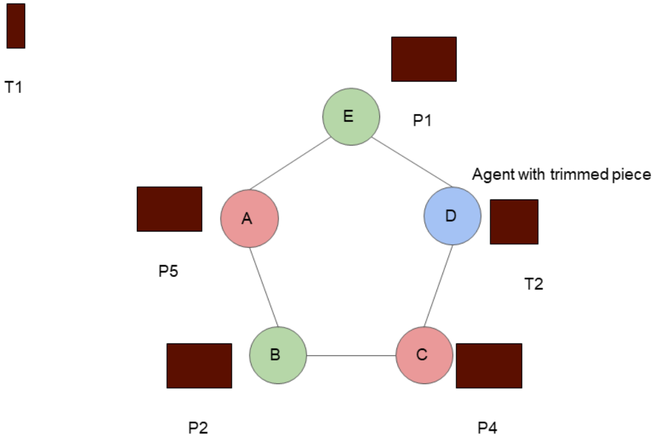
Cycle (C_5)



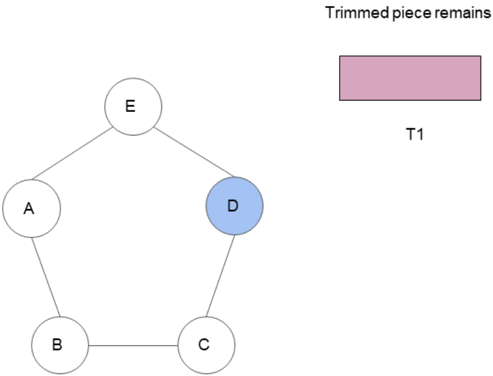
Cycle (C_5)



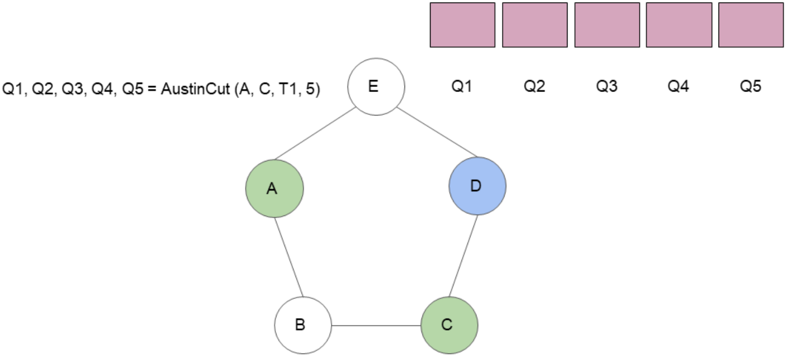
Cycle (C_5)



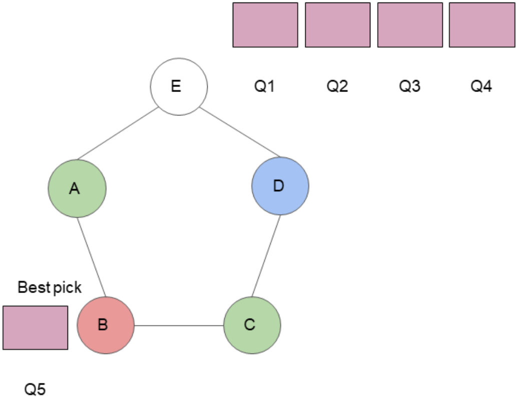
Cycle (C_5)



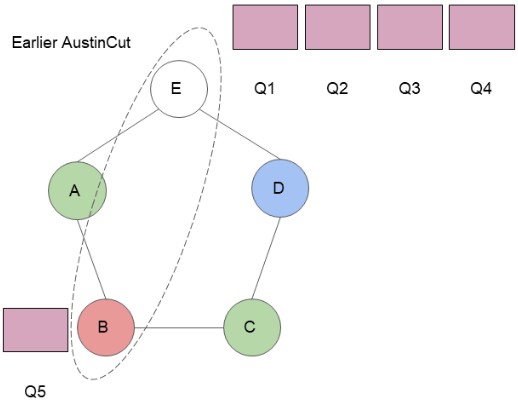
Cycle (C_5)



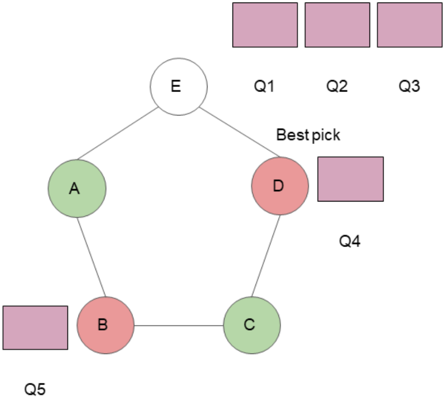
Cycle (C_5)



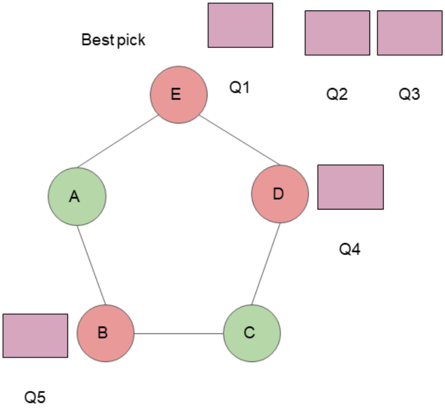
Cycle (C_5)



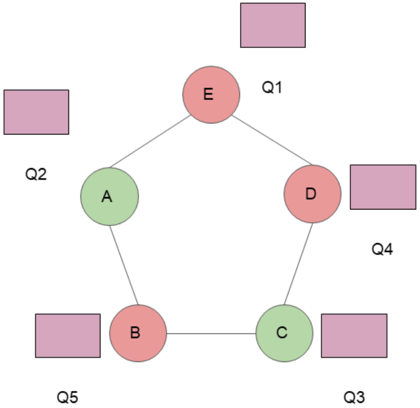
Cycle (C_5)



Cycle (C_5)

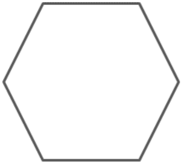


Cycle (C_5)

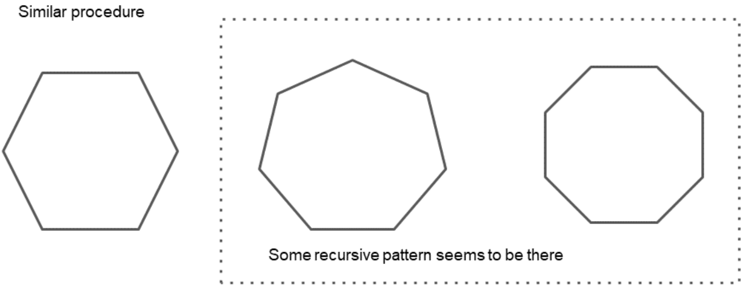


Cycle General

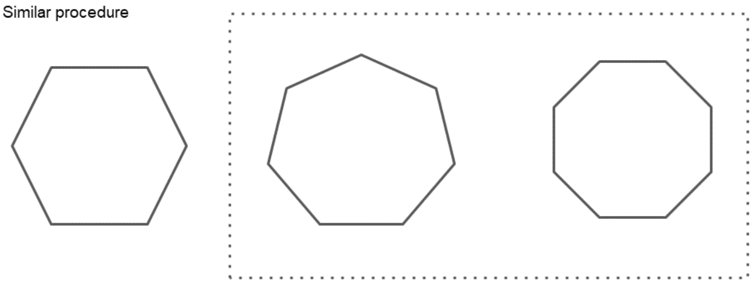
Similar procedure



Cycle General



Cycle General

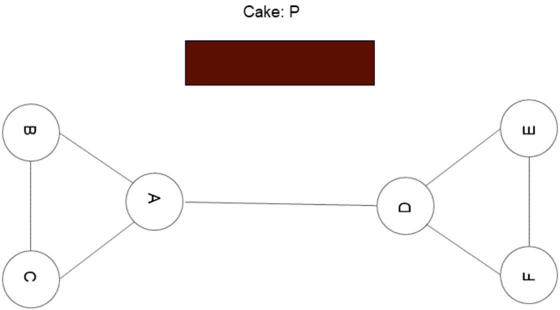


Generalize to N length cycles?

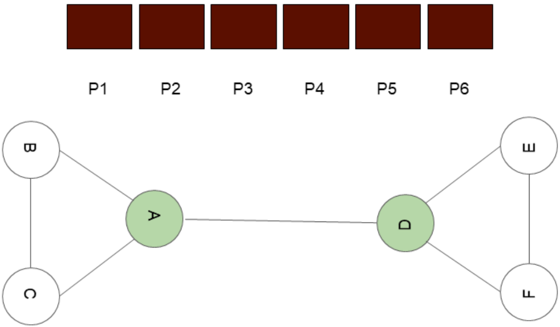
Structures

- Cycles(length: 5, 6)
- Cliques connected via a Bridge (clique with 3 vertices)
- Other Miscellaneous Structures

Clique connected via a Bridge (clique with 3 vertices)

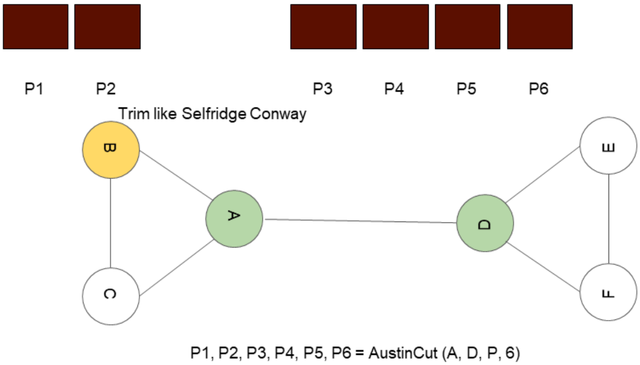


Clique connected via a Bridge (clique with 3 vertices)

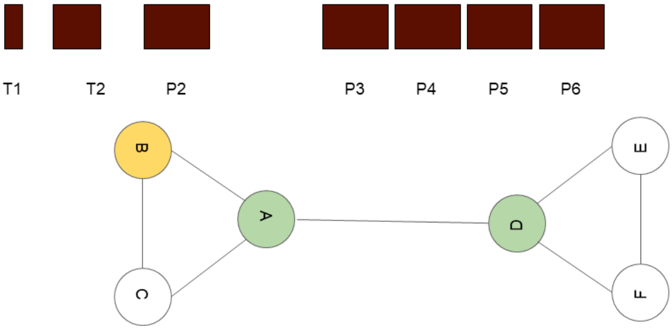


P1, P2, P3, P4, P5, P6 = AustinCut (A, D, P, 6)

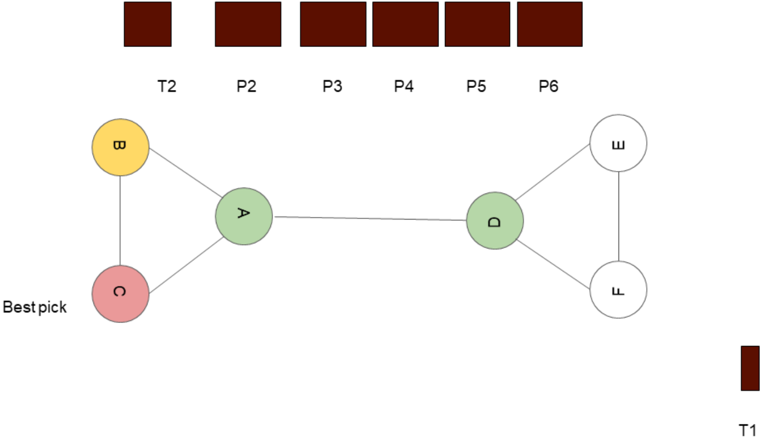
Clique connected via a Bridge (clique with 3 vertices)



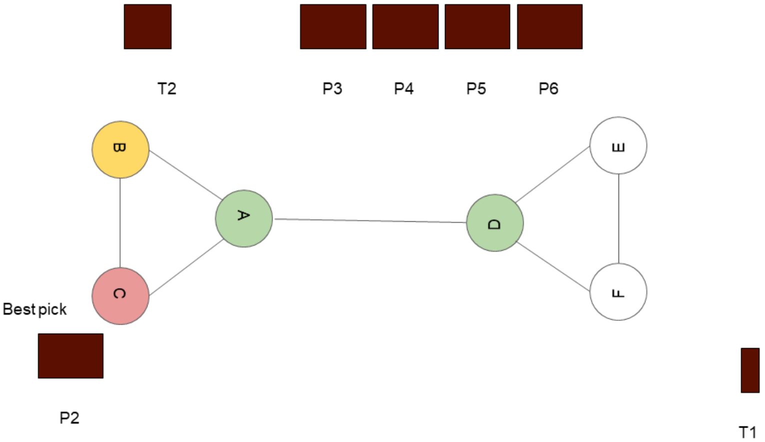
Clique connected via a Bridge (clique with 3 vertices)



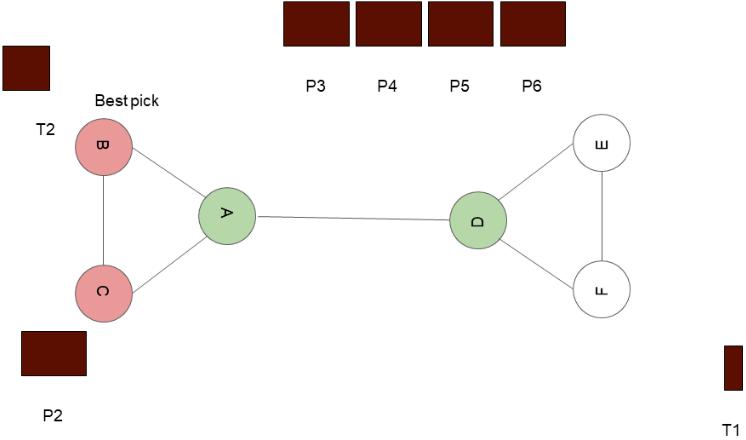
Clique connected via a Bridge (clique with 3 vertices)



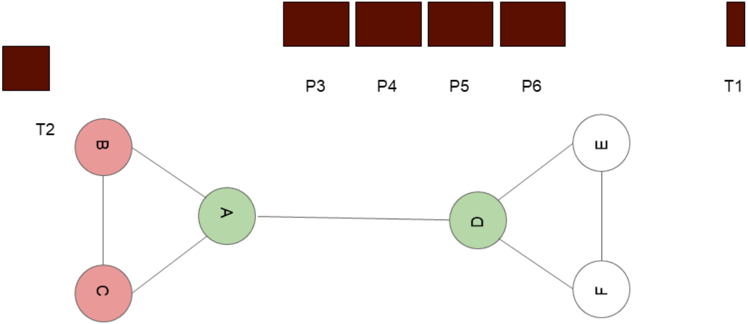
Clique connected via a Bridge (clique with 3 vertices)



Clique connected via a Bridge (clique with 3 vertices)

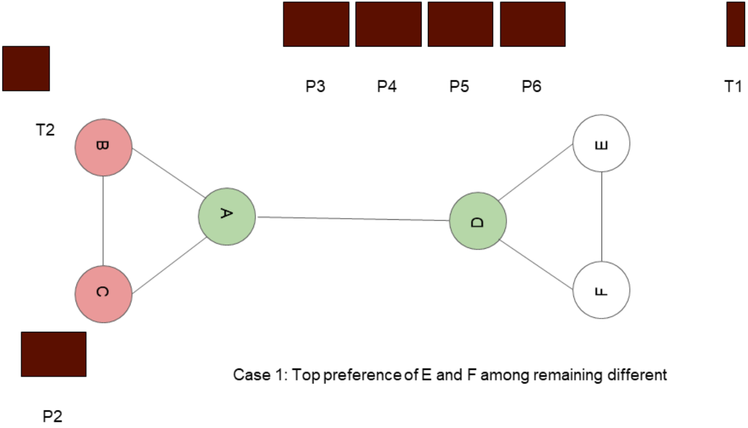


Clique connected via a Bridge (clique with 3 vertices)



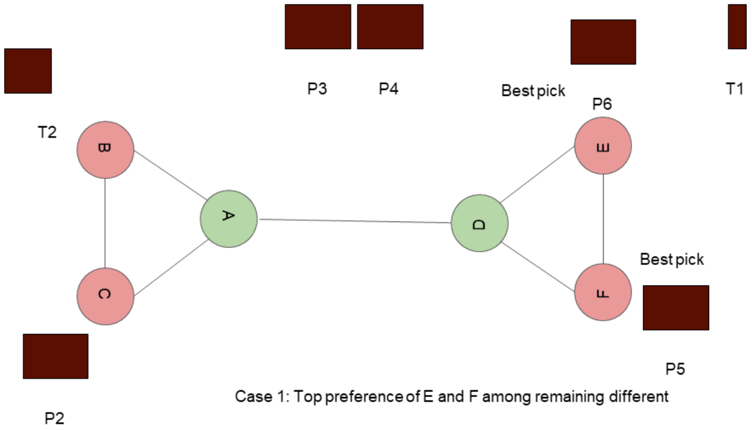
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)

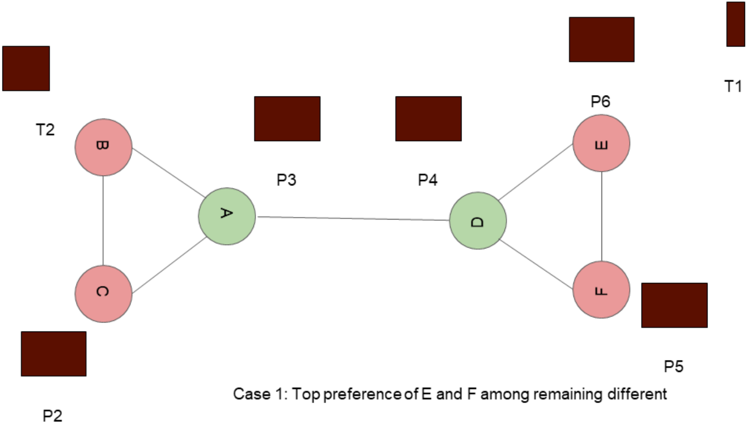


Case 1: Top preference of E and F among remaining different

Clique connected via a Bridge (clique with 3 vertices)

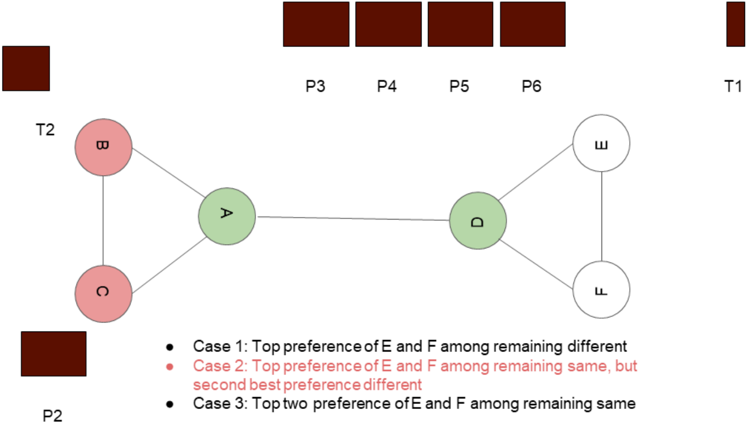


Clique connected via a Bridge (clique with 3 vertices)



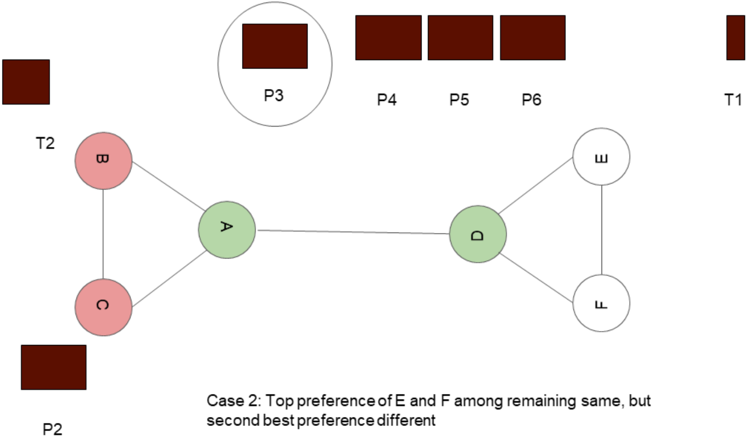
Case 1: Top preference of E and F among remaining different

Clique connected via a Bridge (clique with 3 vertices)



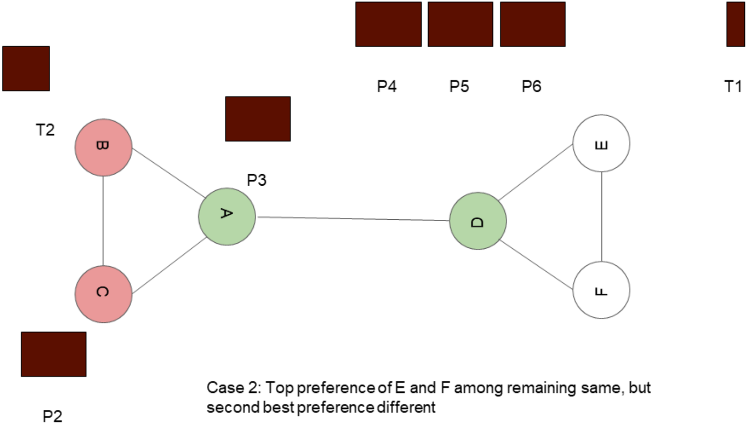
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)



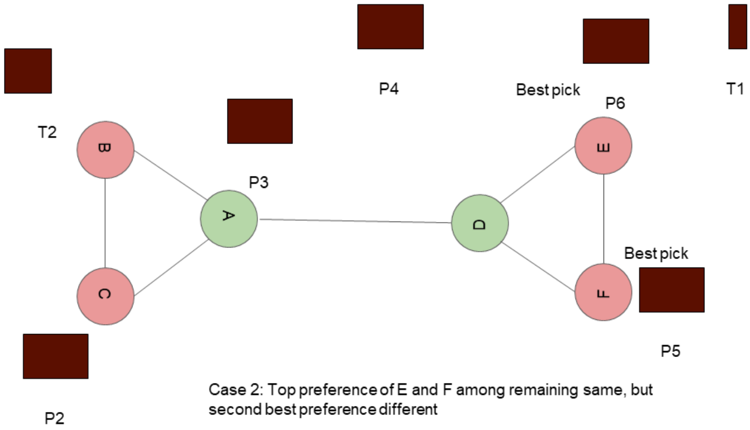
Case 2: Top preference of E and F among remaining same, but second best preference different

Clique connected via a Bridge (clique with 3 vertices)

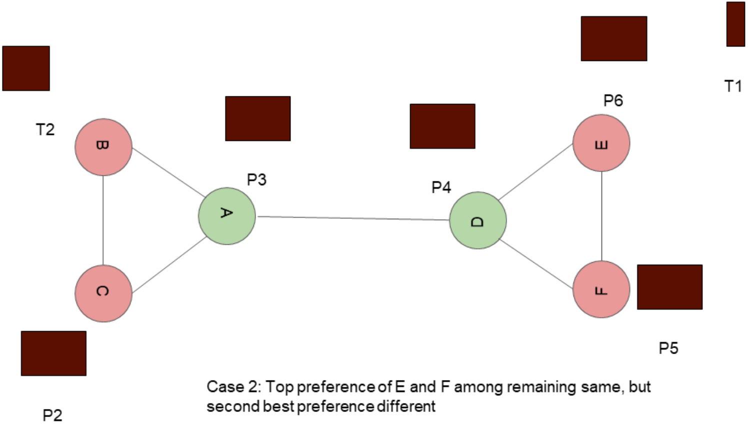


Case 2: Top preference of E and F among remaining same, but second best preference different

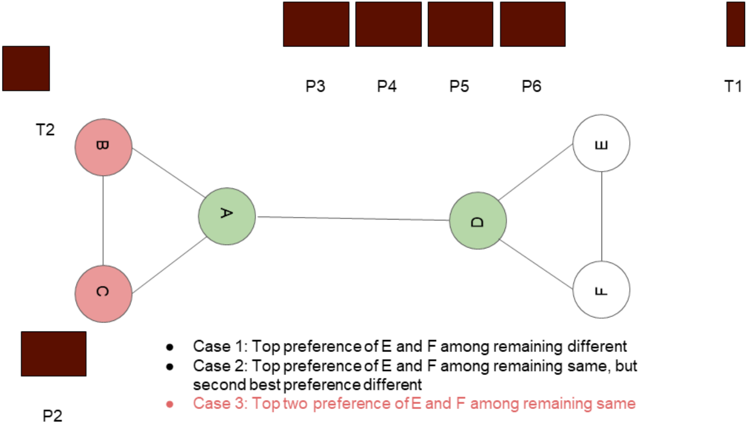
Clique connected via a Bridge (clique with 3 vertices)



Clique connected via a Bridge (clique with 3 vertices)

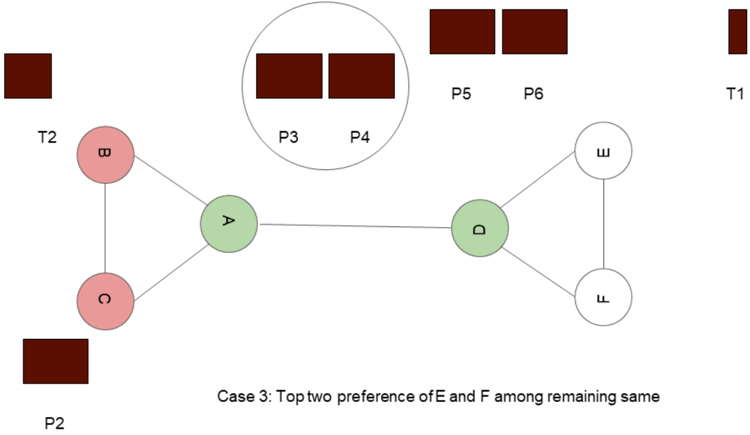


Clique connected via a Bridge (clique with 3 vertices)



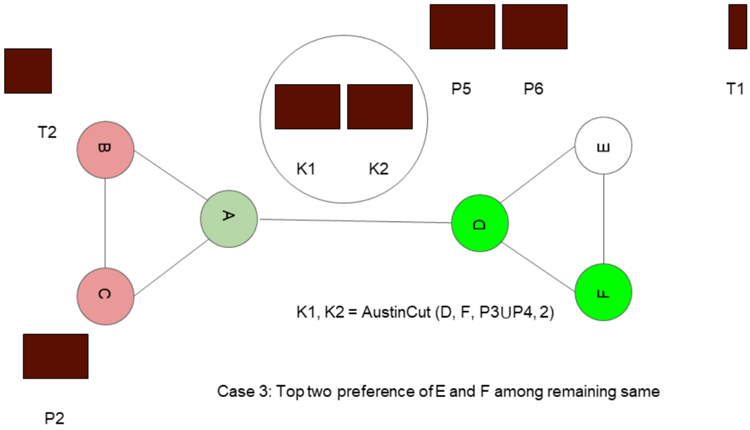
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)

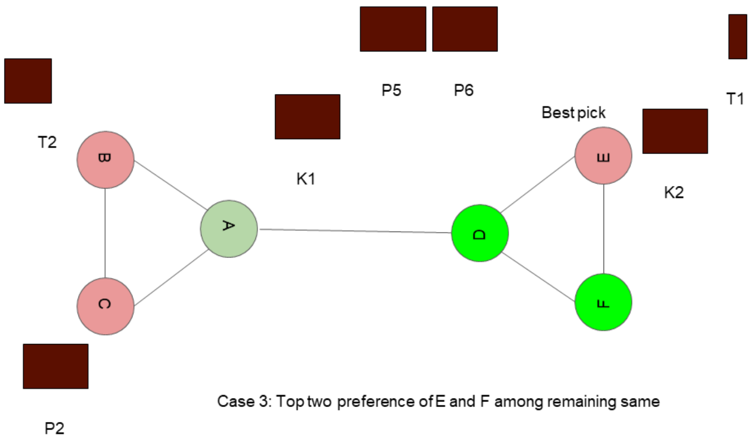


Case 3: Top two preference of E and F among remaining same

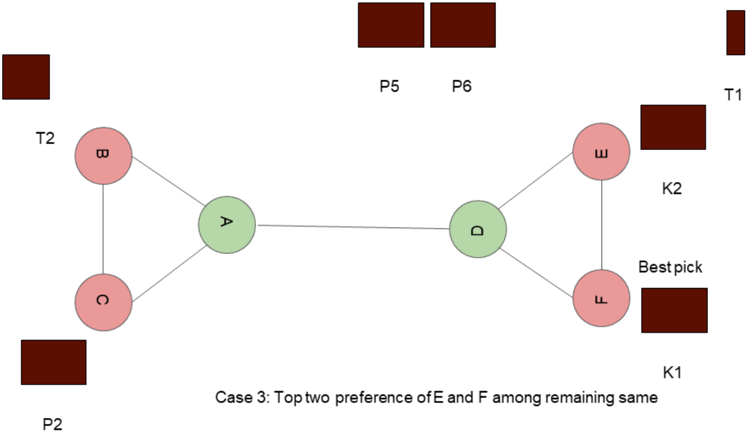
Clique connected via a Bridge (clique with 3 vertices)



Clique connected via a Bridge (clique with 3 vertices)

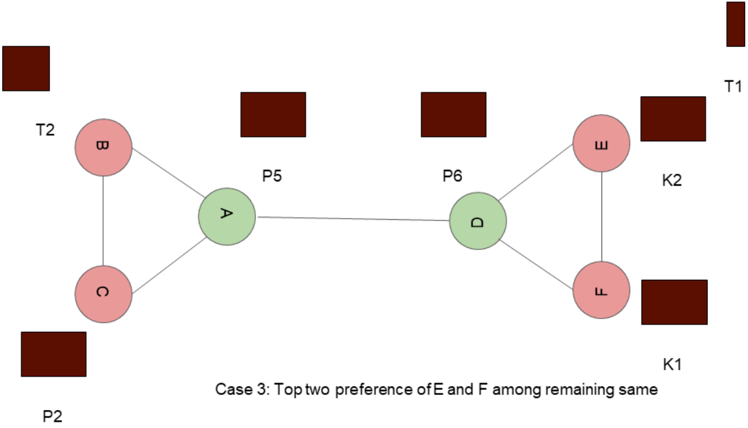


Clique connected via a Bridge (clique with 3 vertices)



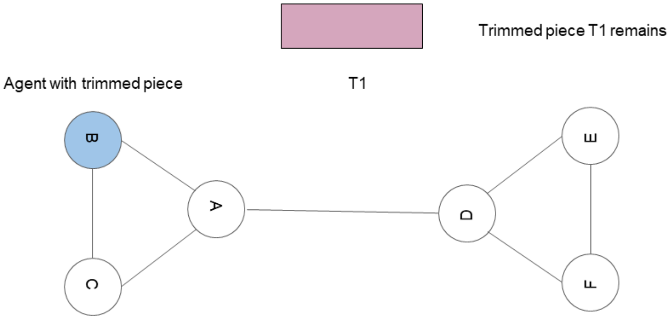
Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)

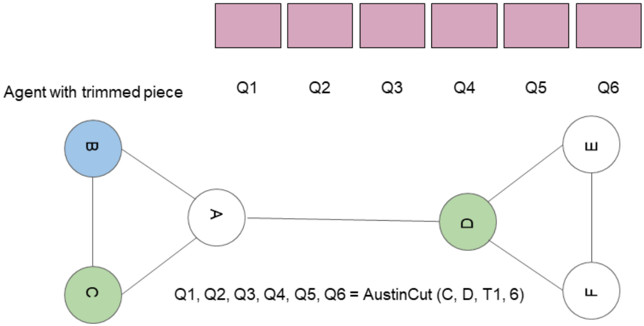


Case 3: Top two preference of E and F among remaining same

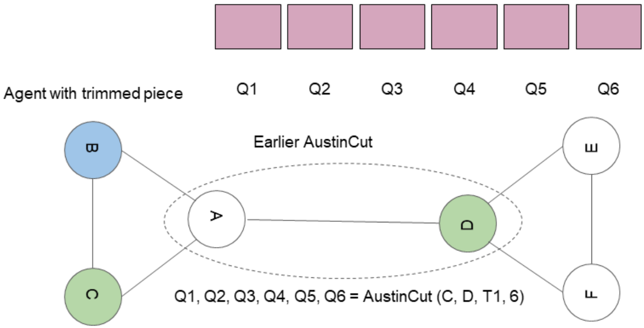
Clique connected via a Bridge (clique with 3 vertices)



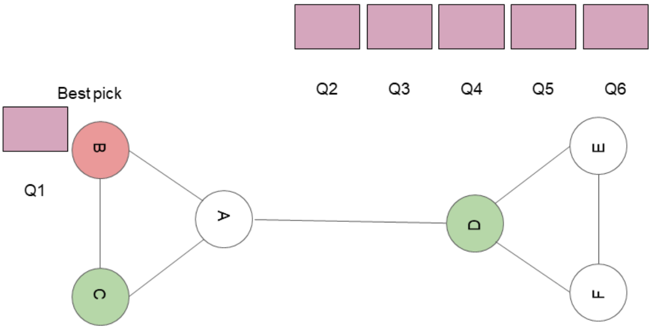
Clique connected via a Bridge (clique with 3 vertices)



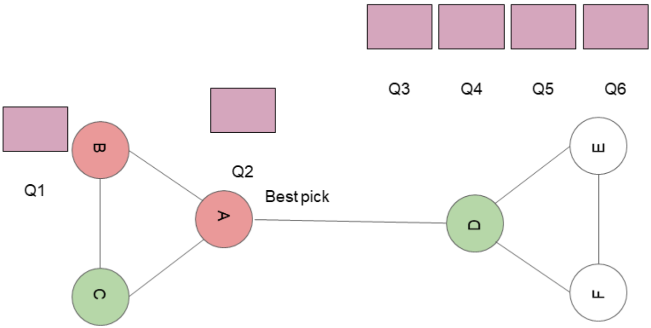
Clique connected via a Bridge (clique with 3 vertices)



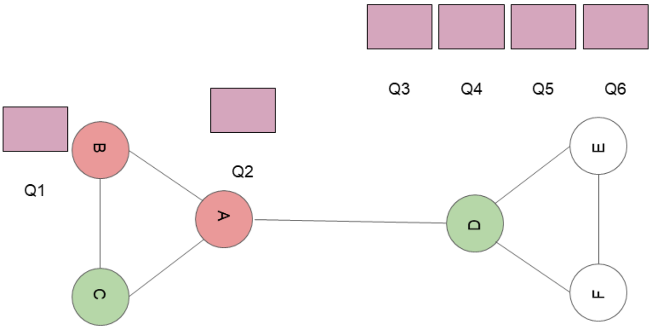
Clique connected via a Bridge (clique with 3 vertices)



Clique connected via a Bridge (clique with 3 vertices)

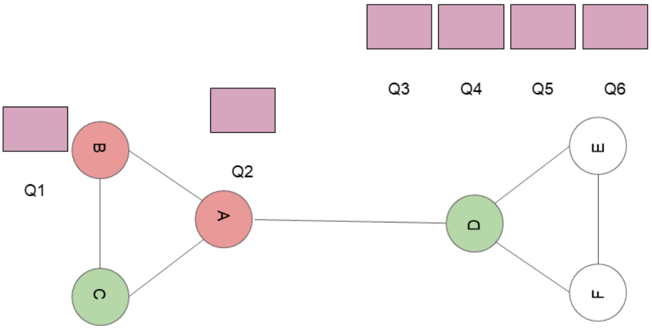


Clique connected via a Bridge (clique with 3 vertices)



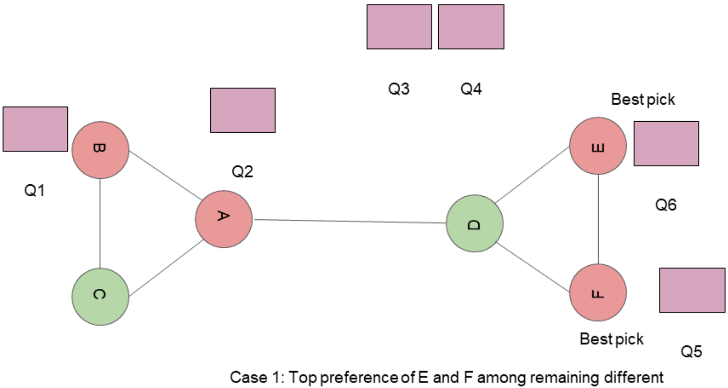
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)

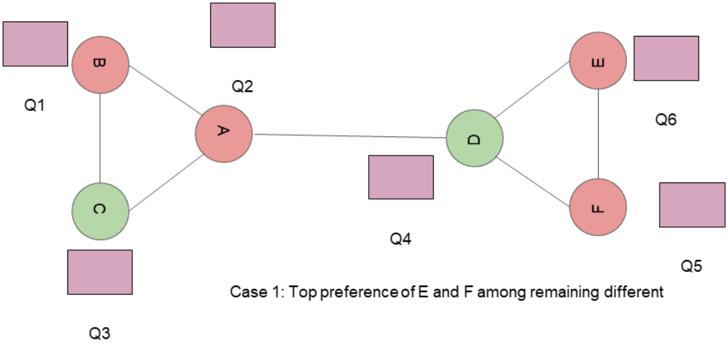


Case 1: Top preference of E and F among remaining different

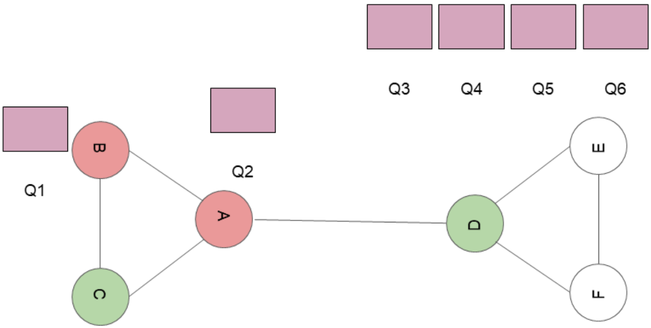
Clique connected via a Bridge (clique with 3 vertices)



Clique connected via a Bridge (clique with 3 vertices)

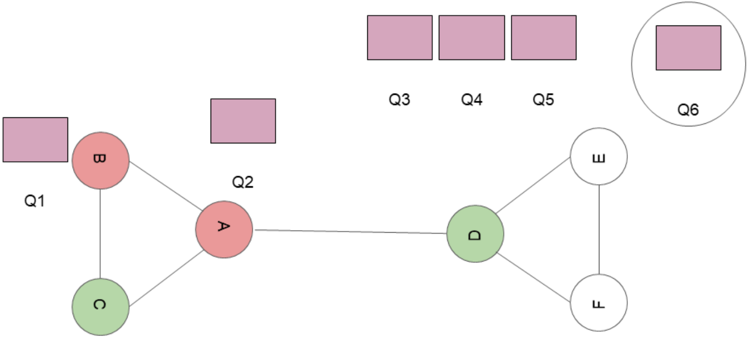


Clique connected via a Bridge (clique with 3 vertices)



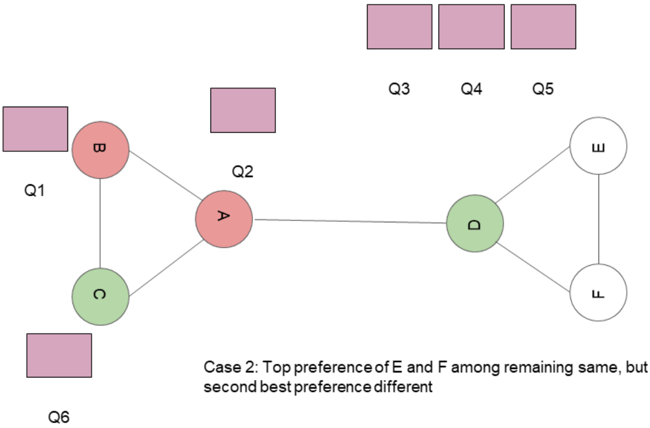
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)

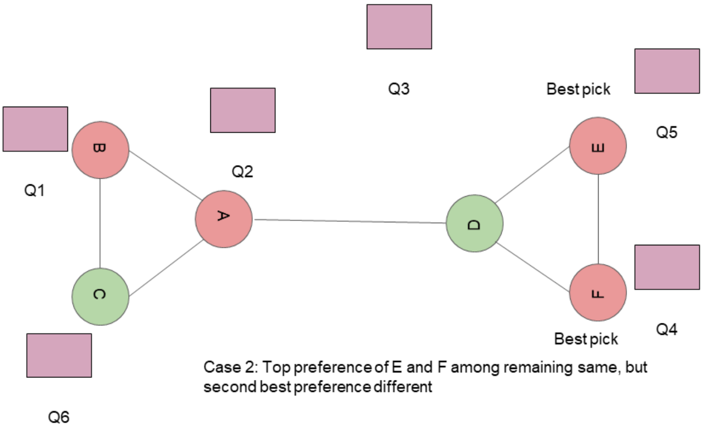


Case 2: Top preference of E and F among remaining same, but second best preference different

Clique connected via a Bridge (clique with 3 vertices)

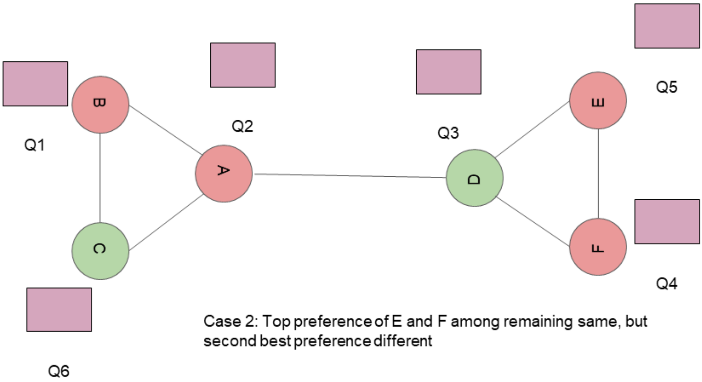


Clique connected via a Bridge (clique with 3 vertices)



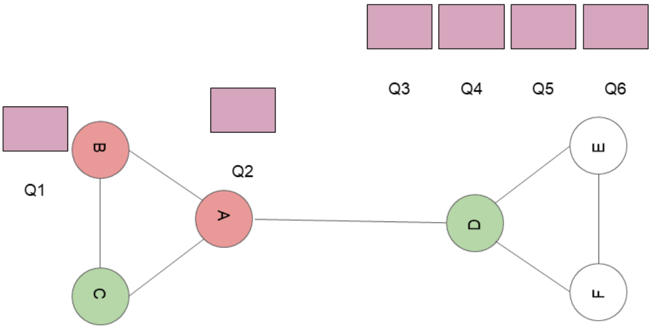
Case 2: Top preference of E and F among remaining same, but second best preference different

Clique connected via a Bridge (clique with 3 vertices)



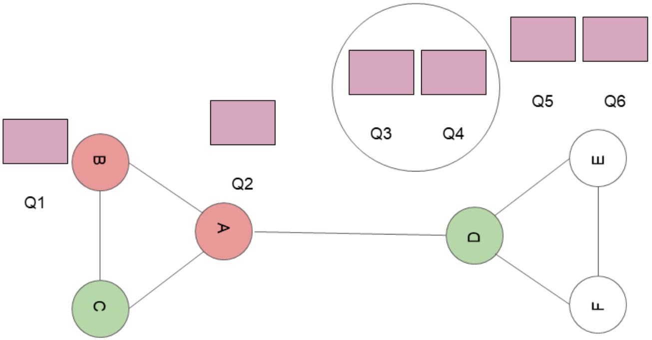
Case 2: Top preference of E and F among remaining same, but second best preference different

Clique connected via a Bridge (clique with 3 vertices)



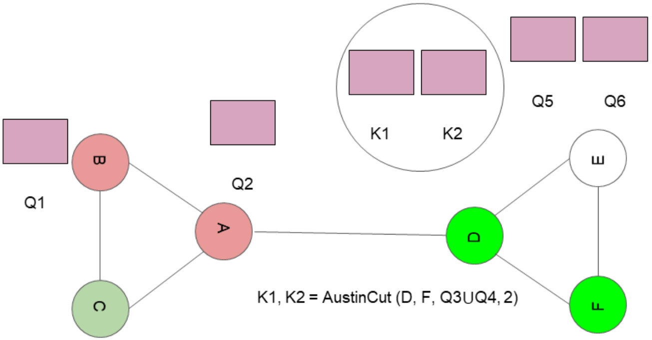
- Case 1: Top preference of E and F among remaining different
- Case 2: Top preference of E and F among remaining same, but second best preference different
- Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)



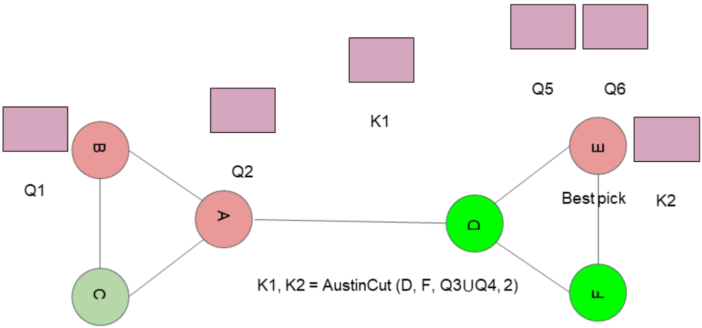
Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)



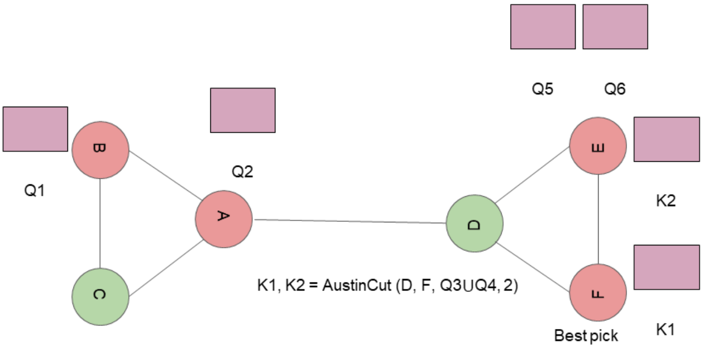
Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)



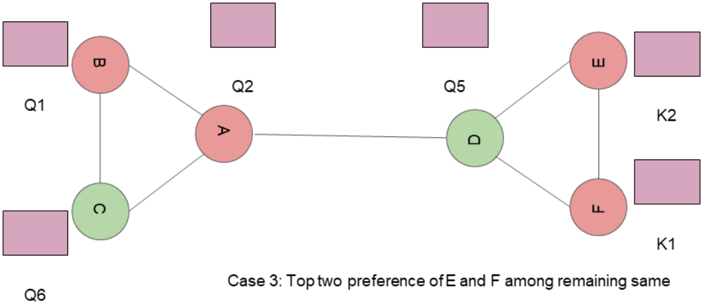
Case 3: Top two preference of E and F among remaining same

Clique connected via a Bridge (clique with 3 vertices)



Case 3: Top two preference of E and F among remaining same

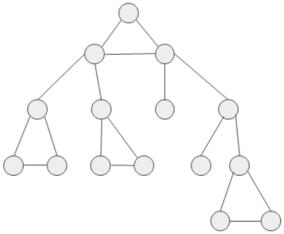
Clique connected via a Bridge (clique with 3 vertices)



Extension

Possible extensions of this structure

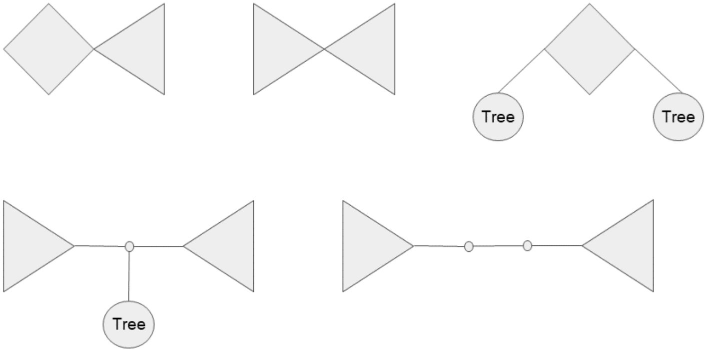
Adding more edges to a tree, particularly
leaf sibling edge to a binary tree



Structures

- Cycles(length: 5, 6)
- Cliques connected via a Bridge (clique with 3 vertices)
- Other Miscellaneous Structures

Other Miscellaneous structures



What's Next?

Possible Future Directions:

- *General algorithm for C_n* - For cycle networks, we have algorithms upto C_6.
- *Extensions to* [BQZ17] - Our extension only added 1 edge. We have an algorithm for adding leaf edges, but it involves wastage.
- *Limitations of continuous procedures* - It isn't known how far these processes can bring us in obtaining efficient algorithms. We can try finding discrete algorithms, but even for simpler structures like C_4, they have the possibility of being very complicated.
- *Other fairness notions* - Proportionality is generally easier to obtain than envy-freeness, but is harder to generalize over several networks. Nevertheless, we can try to obtain proportional and efficient algorithms. We can also try to define and study other notions of networked fairness.

Our References

[AM16a] Haris Aziz and Simon Mackenzie. "A discrete and bounded envy-free cake cutting protocol for any number of agents". In: *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE. 2016, pp. 416–427.

[AM16b] Haris Aziz and Simon Mackenzie. "A discrete and bounded envy-free cake cutting protocol for four agents". In: *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*. 2016, pp. 454–464.

[BQZ17] Xiaohui Bei, Youming Qiao, and Shengyu Zhang. "Networked fairness in cake cutting". In: *arXiv preprint arXiv:1707.02033* (2017).

[BTZ97] Steven Brams, Alan Taylor, and William Zwicker. "A moving-knife solution to the four-person envy-free cake-division problem". In: *Proceedings of the american*

Conclusion

Thank you!