Seating Arrangement Problem:

Sorting Out Children by Sorting Out Digraphs

By Tony Leguia

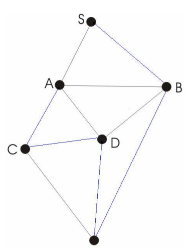
Problem:

We examined the problem of seating a classroom of students given a set of preconditions. The teacher has each child provide the names of two enemies (ξ) and two friends (φ). A solution to this problem can be found by modeling the children and their relationships as a directed graph with two kinds of edges. One type of edge represents the "friend" relationship and another type of edge represents the "enemy" relationship. The indifferent relations are demonstrated by the complement of the graph, or can be expressed by no shared edges between vertices, or by a third type of edge. A variation of the original problem would be to parameterize the number of friends and enemies and attempt to characterize the properties of such graphs.

Definitions:

- A graph is a collection of points or vertices, connected by lines or edges.
- If these edges have direction then the graph is a directed graph, or digraph. Edges can also have weights which define cost.
- A path is a sequence of vertices such that each vertex is visited once.

Example: Non-directed graph with Hamilton Path shown in blue.

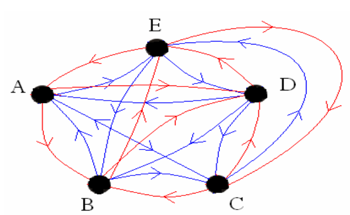


Example:

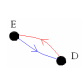
To illustrate our method we have the following example, red edges indicate "enemy" relationships, while blue edges indicate "friend" relationships.

To find a proper seating arrangement for a one-dimensional classroom let us start at vertex E in digraph G. We must pick one friend or a vertex with which E does not share any edges. Pick D. Notice D does not like E, therefore the next vertex must share a friend edge D. Pick C. Likewise C does not like D, therefore the next vertex must be a friend of C. Next choose A as it shares a friend edge with C. Notice that A and C share a reciprocal relationship, therefore any vertex may follow, so long as it likes A, or is indifferent to it. Finally place B at the end as the arc (B, A) is a friend arc. Our final seating arrangement for this graph is [E][D][C][A][B].

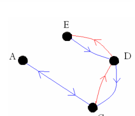
Digraph G



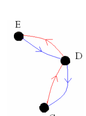
Step 1.



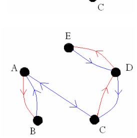
Step 3.



Step 2.



Step 4.

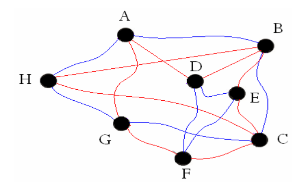


Process:

To find a solution to this problem for any graph G(V, E) with n vertices, we began by simplifying the problem and solving the simplified version. We then started removing restrictions and gradually increased the complexity of the problem. We wrote programs to find solutions to graphs and then analyzed the problem to arrive at conclusions of the characterizations of these types of graphs.

Problem 1: Reciprocal Relationships

- Assume every relationship is reciprocal. That is, if child A likes child B then B likes A, and the same goes for dislike relationships. Therefore we do not require a digraph to represent this problem.
- Make the class one-dimensional, each child has a position, with children before and after it.

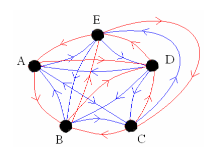


Solution to Problem 1:
Start at any vertex A, go to a friend: call it B. B must share an edge with a third unvisited vertex C. Go to C. C might share an edge with A, creating a cycle, otherwise it will share a vertex with an unvisited vertex. If stuck within a subgraph that results in a cycle (V= E, F, D in the example), move to another subgraph. Repeat until a solution is found.

Problem 2: Relationship Dynamics

The next step was to remove the symmetry of relationships. Now If vertex A likes vertex B, B is allowed to hate A. Because of this digraphs are now needed to properly model the problem. Doing this adds complexity because now the number of possible edges has increased, and the dynamics of relationships has become more complex, i.e. $b \rightarrow a$ but $\neg(a \rightarrow b)$ is now a legal configuration.

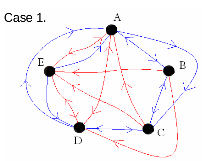
Example:



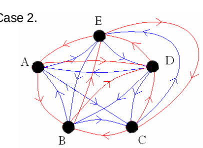
This problem results in several special cases that require attention:

Case 1: What if each vertex in G hates one particular vertex, E?
Case 2: What if each vertex that a given vertex likes, hates it back?

Case 1.



Case 2.



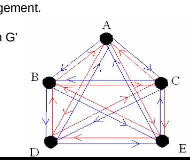
Case 1 is interesting if for any seating arrangement you must place E at the end of the arrangement and the previous node is not content.

Case 2 is more complex. The last vertex in a seating arrangement for this type of graph will never be happy, since it is impossible for it to be near a friend. Therefore it must neighbor a vertex to which it is indifferent. This may be impossible for smaller graphs.

Results:

- Each vertex has n^2 -5n indifferent relationships
- We found that there did not exist a solution for graph G'.
- Every graph G(V, E) with V size 5, and for some a,b in the set of V such that $a \leftrightarrow b$ has a seating arrangement.

Graph G'



Problem 3: Two-dimensions Introduced

We cease to think of the classroom as a one dimensional construct, instead we use a matrix as our model. This adds new complexity to the problem because now we must check all vertices surrounding a given node, not just the vertices preceding and following it. Similarly, this gives a given vertex more opportunities to be happy since it's state can be affected by any of the surrounding vertices. Additionally, it becomes much more difficult to surround a vertex with vertices it is indifferent to, except in very large graphs.

Example of a two-dimensional seating arrangement.

[A][G][J][E]
[B][F][I][D]
[C][H][L][K]

Vertices F and I are affected by four other vertices. G, J, B, D, H, and L can be affected by three vertices. Only the remaining four vertices have only two surrounding vertices to worry about.

Results:
All several hundred of the two-dimensional classrooms that we tested had legal seating arrangements.

Problem 4: Parameterized Relationships

We now parameterize the number of friends and enemies, allowing a vertex any number of enemies and friends so long as they add up to no more than n-1. We also allow additional preconditions to be placed on the placement of the children. Restrictions allowed include mandatory placement locations and restrictions on neighbors. This new problem leads to some special cases that required our attention.

Case 1: 2 friends; n-3 enemies
Case 2: no friends, each vertex has at least two indifferent relationships
Case 3: Only one friend per vertex

Change use of the word problem to obstacle etc…..

Case 1 is an obstacle since every vertex must be positioned next to one friend. Especially problematic for large graphs where multiple vertices are friends with the same node.

Case 2 is interesting because this case has no solution if a vertex exists such that every other vertex hates it. For small graphs it may prove difficult, if not impossible, to surround a vertex with other vertices which share reciprocal indifference relationships with it.

Case 3 graphs will not have a seating arrangement if there exists a vertex that every other vertex shares a "enemy" edge with.

Results:

- If $\varphi > \xi$ then a seating arrangement exists
- If size(φ)= 1then G has a seating arrangement if there exists a relationship such that for any a,b Є V : $a \rightarrow b \wedge b \rightarrow a$

Problem 5: Mandatory Seating

The final addition to the problem was to allow the user to define additional preconditions to the seating arrangement. Possible restrictions are:

Mandatory seating location- vertex must sit in this location

Seating location restrictions- vertex cannot sit in this location

Enforced neighbor- vertex must sit next to specified vertices

Prohibited neighbor- vertex cannot next to specified vertices

Because of the nature of allowing the user to define the restrictions and configurations requirements, it is feasible to enter a configuration that is impossible to develop. For example, if one prohibits a vertex from sitting next to any of its friends, and there are not enough vertices to which this vertex is indifferent towards, there would be no possible seating arrangement.

Conclusions and Future Work:

At the moment our program looks at all possible arrangements of the vertices in G. This gives our program a worst case runtime of O(n!). It might be possible to optimize, or provide the program with heuristics that allow for a more time efficient runtime. We would also like to develop formal proofs for our conjectures. Additionally, we would like to further characterize how restrictions and seating requirements (Problem 5) affect the seating arrangements of graphs.

Acknowledgements:

Dr. Sean McCulloch for his help, advice and mentoring.

Ohio Wesleyan University for providing this research opportunity and the facilities and resources that made this research possible.

The NSF for providing the funding that made this research possible.