

ASSIGNING SCOUTS TO OPTIMAL PATROLS

PHIL SNYDER, IAN BRUCE, AND JULIO MARCO PINEDA

ABSTRACT. Scoutmaster Gene Bruce of the Boy Scouts of America for Troop 407 in Kent, Washington wants to find an arrangement of 13 new scouts into two patrols. The cohesiveness and overall quality experience of a patrol is affected by adverse relationships and existing friendships. By using anonymous data from scouts and parents, a brute force strategy was employed to find an optimal arrangement of scouts that avoids severe conflict while maximizing pre-existing preferences.

PROBLEM DESCRIPTION

The community partner we are serving for this project is Scoutmaster Gene Bruce of the Boys Scouts of America. He leads Troop 407 in Kent, Washington and every year he must assign new scouts to groups - called “patrols”. In these patrols, scouts perform different scouting activities together and form close bonds. However, in recent years, the number of scouts have been increasing at such a rate that it is no longer logistically viable to place every new scout in a single patrol; multiple patrols are needed to organize activities better and to form a community between new recruits. Furthermore, scouts are in the fifth or sixth grade and have not yet fully developed their interpersonal skills. Many can be temperamental and awkward when dealing with adversity. Devastating conflicts may occur if two at-odds scouts create animosity between each other, alienating themselves from the rest of the patrol. If enough of these adverse interactions occur within a patrol, scouts may decide to drop out of the Boy Scouts, losing out on a potentially great experience and community of friends.

The data the Scoutmaster collects about adverse pairings and preferences are either privately given by the scouts themselves or by their parents. The parents preferences are hard rules that we must abide by (if possible). In the past, the Scoutmaster determined the perfect arrangement of patrols without too much time and effort due to the smaller number of new recruits. However, newer scouts are joining yearly to the point that solving this grouping problem cannot be done by hand in a reasonable amount of time. Therefore, we can help our community partner find a better means to solve his troop organization problem. Our goal is to assign boy scouts in different patrols of appropriate size (of 6-8 scouts) such that we maximize the retention rate of the scouts in the program by avoiding severe conflicts while maximizing positive relationships between scouts.

We aim to answer the particular instance of this problem provided by our community partner, that of arranging 13 scouts into two troops that avoids severe conflicts while promoting positive relationships between scouts. Furthermore, we want to determine if we can build a model to determine optimal patrol arrangements for any number of scouts so that the Scoutmaster can use this model for future years. For 13 scouts, a grouping into one patrol is too large, while three patrols are too

A hand-drawn network diagram on lined paper showing relationships between 12 people: Jordan, Brandon, Darwin, Christian, Jake, Cameron, Timmy, Colby, Daniel, Patrick, Nathan, and Evan. Nodes are circles, and edges are arrows. Some edges are solid, some dashed, and some are thick red.

Nodes (People):

- Jordan
- Brandon
- Darwin
- Christian
- Jake
- Cameron
- Timmy
- Colby
- Daniel
- Patrick
- Nathan
- Evan

Relationships (Edges):

- Thick Red Edges:**
 - Jordan to Cameron
 - Jordan to Timmy
 - Nathan to Timmy
- Solid Edges:**
 - Brandon to Jordan
 - Brandon to Darwin
 - Darwin to Christian
 - Christian to Jake
 - Jake to Daniel
 - Daniel to Patrick
 - Patrick to Nathan
 - Nathan to Evan
 - Evan to Cameron
 - Cameron to Timmy
 - Timmy to Colby
 - Colby to Daniel
 - Daniel to Patrick
- Dashed Edges:**
 - Jordan to Brandon
 - Brandon to Jordan
 - Brandon to Darwin
 - Darwin to Brandon
 - Darwin to Christian
 - Christian to Darwin
 - Christian to Jake
 - Jake to Darwin
 - Jake to Daniel
 - Daniel to Jake
 - Daniel to Patrick
 - Patrick to Daniel
 - Patrick to Nathan
 - Nathan to Patrick
 - Nathan to Evan
 - Evan to Nathan
 - Evan to Cameron
 - Cameron to Evan
 - Cameron to Timmy
 - Timmy to Cameron
 - Timmy to Colby
 - Colby to Timmy
 - Colby to Daniel
 - Daniel to Colby
 - Daniel to Patrick
 - Patrick to Daniel

The Stirling number of the second kind provides the total possible ways to arrange any number of distinguishable objects to indistinguishable groups as long as any troop size is permitted (except the empty troop). For the data in this project, the Stirling number of the second kind for 13 scouts into 2 troops is 4095^2 . This number provides an upper bound on the possible arrangements for the specific problem tackled in this project.

Ultimately, our goal is to find groups for the scouts that will allow them to enjoy their time scouting to the fullest. Modeling interpersonal relations by likes and dislikes is, of course, a simplification. But the difficulty in modeling the intricacies

²See Appendix B

of human relationships demands it. In addition, our process will have to be repeated each year for every new batch of scouts, and would perhaps be too burdensome to warrant the effort if our process attempted to delve too far into the psyche.

Another simplification is to assume that the quality of a group can be determined by the quality of relationships between unique pairs either inside or outside the group - in other words, the whole is equal to the sum of the parts. For example, consider the case where scouts A, B and C individually like each other when they are alone with one of their group, but don't like being in a group together. This type of epiphenomenon won't be considered.

MATHEMATICAL MODEL

To model the relationships between scouts, we use a mathematical graph. We let the scouts be vertices in our graph. A directed edge between Scout A and Scout B has a weight proportional to how much Scout A likes Scout B. We would then like to find a partition of the vertices in the graph that either minimizes or maximizes an objective function given constraints on the sizes of subsets generated by our partition. This results in a complete, directed graph on 13 vertices having edge weights 0, 1, or -1, corresponding to indifference, approval, or disapproval, respectively. Scoutmaster Bruce specifically would like two patrols from the 13 scouts - one of size 6, the other of size 7. The total number of partitions given these constraints is $\binom{13}{6} = \binom{13}{7} = 1716$. This number is reasonable enough that we may calculate the total loss of every valid grouping and select the partition with the lowest or highest score depending on the objective function.

We measure the goodness (or score) of a partition in six different ways, corresponding to six different objective functions, which we dub MinCut, FriendCut, EnemyCut, AwkwardCut, and HybridCut. MinCut is designed to minimize the cut edge weights between partitions (see *Solution of Mathematical Problem*). FriendCut is designed to maximize the number of intra-patrol positive edge weights. EnemyCut is meant to minimize the number of intra-patrol negative edge weights. Awkward cut (+ i) is the same as MinCut, but with an additional penalty of i if there are a pair of scouts, Scout A and Scout B, such that Scout A favors Scout B, but Scout B disfavors Scout A. HybridCut is a convex combination of MinCut and FriendCut, each with equal weight (that is, half the MinCut loss plus half the FriendCut loss). Our three primary objective functions - MinCut, FriendCut, and EnemyCut are given a more rigorous treatment in the next section.

SOLUTION OF MATHEMATICAL PROBLEM

We represent our directed graph as a 13 by 13 similarity matrix, or, as might be more fitting for the problem, an "affinity" matrix. Like below³:

The NA value represents a hard constraint established by a scout's parent (i.e., under no circumstances are these two to be in the same patrol). We designed a number of objective functions that accept two scouts and calculates the loss associated with such a placement.

³The scouts were enumerated like so: 1. Brandon, 2. Cameron, 3. Christian, 4. Colby, 5. Daniel, 6. Darwin, 7. Evan, 8. Jake, 9. Jordan, 10. Nathan, 11. Patrick, 12. Timmy, 13. Tommy.

1	0	1	-1	0	0	0	0	0	1	-1	0	0	1
2	0	0	-1	1	1	0	0	0	0	0	0	0	1
3	0	0	0	0	0	-1	0	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0	NA	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	1	1	0	0	1	0	0	0	0	0
7	0	1	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	1	1	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	1	0	0	0	1	0
2	0	0	0	0	1	0	0	0	0	1	0	0	1
3	-1	1	0	1	0	0	0	0	-1	0	0	1	0

RESULTS

As mentioned earlier, the number of scouts we must partition is small enough to find an optimal solution by brute force. Our algorithm finds a partition of scouts into patrols of sizes 6 and 7 which is optimal with respect to each objective.

MinCut		FriendCut		EnemyCut	
Patrol 1	Patrol 2	Patrol 1	Patrol 2	Patrol 1	Patrol 2
Brandon Cameron Colby Darwin Evan Tommy	Christian Daniel Jake Jordan Nathan Patrick Timmy	Brandon Cameron Christian Colby Evan Tommy	Daniel Darwin Jake Jordan Nathan Patrick Timmy	Brandon Cameron Colby Daniel Darwin Evan Tommy	Christian Jake Jordan Nathan Patrick Timmy
AwkwardCut (+1)		AwkwardCut (+2)		HybridCut	
Patrol 1	Patrol 2	Patrol 1	Patrol 2	Patrol 1	Patrol 2
Brandon Cameron Colby Darwin Evan Tommy	Christian Daniel Jake Jordan Nathan Patrick Timmy	Brandon Christian Daniel Darwin Jake Jordan Patrick	Cameron Evan Colby Nathan Timmy Tommy	Brandon Cameron Colby Evan Timmy Tommy	Christian Daniel Darwin Jake Jordan Nathan Patrick

FIGURE 1. The partitions arrived at by minimizing each objective function

For each method we define three scoring metrics:

- (1) **MinCut**. This is the traditional metric of a cut on a weighted graph and is the sum of the weights of the edges we would “cut” if we were to sever the edges between distinct patrols. More formally, for some partition I, J $I \neq J$, the MinCut is defined as:

$$\text{MinCut}(I, J) := \sum_{e=(i,j), i \in I, j \in J} c(e)$$

We want this to be as low as possible.

- (2) **FriendCut**. This is the sum of the positive, or friendly edge weights within each patrol. That is, if $I_+ = \{e = (i, j) | i, j \in I, c(e) = 1\}$ is the set of edges in patrol I with positive weight and $J_+ = \{e = (i, j) | i, j \in J, c(e) = 1\}$ is the set of edges in patrol J with positive weight, then (assuming all positive weights are 1):

$$\text{FriendCut}(I, J) := |I_+ \cup J_+|$$

We want this to be as high as possible.

- (3) **EnemyCut**. This is similar to FriendCut, except we now measure the number of negative edges going from one scout to another within the same patrol. Let $I_- = \{e = (i, j) | i, j \in I, c(e) = -1\}$ be the set of edges in patrol I with negative weight and $J_- = \{e = (i, j) | i, j \in J, c(e) = -1\}$ be the set of edges in patrol J with negative weight, then

$$\text{EnemyCut}(I, J) := |I_- \cup J_-|$$

We want this to be as low as possible.

The other methods are modifications of these three and are defined in Appendix A.

Method	MinCut	FriendCut	EnemyCut
MinCut	0	17	0
FriendCut	1	18	1
EnemyCut	2	15	0
AwkwardCut (+1)	0	17	0
AwkwardCut (+2)	1	18	2
HybridCut	0	18	1

FIGURE 2. The scores according to the three primary metrics.

Of the objective functions we chose to use, it seems like MinCut and HybridCut are the best performing. FriendCut performs almost as well as HybridCut, except for one worse in the EnemyCut objective. EnemyCut does poorly since finding a partition with 0 intra-patrol negative weight edges is easy in this example, and so EnemyCut settles for the first suboptimal (in terms of the other objective functions) solution it finds. AwkwardCut (+1) arrived at the same partition as MinCut (not altogether surprising, but informative, since even with the additional +1 penalty for placing Brandon and Tommy in the same patrol it decided it would be best off to do so anyways). But changing the “awkward” penalty to +2, the same penalty as mutual dislike in EnemyCut, we get a completely different partition.

Let’s take a more critical look at the optimal partition arrived at via MinCut:

TABLE 1. The partition arrived at by optimizing MinCut

Patrol 1	Patrol 2
Brandon, Cameron, Colby, Darwin, Evan, Tommy	Christian, Daniel, Jake, Jordan, Nathan, Patrick, Timmy

- (1) Tommy and Timmy both like each other and are in separate patrols. But if we move Timmy to Patrol 1 ($P1$), we break up the mutual friendship of Timmy and Nathan in $P2$. Suppose we moved both Timmy and Nathan to $P1$ and Brandon (who dislikes Nathan) to $P2$. But then Brandon loses two friends (Tommy and Cameron), gains one (Jordan), and now neither Nathan nor Timmy are in the same patrol as Daniel, whom they both like. Although, Brandon and Tommy find themselves in the awkward situation of having complete opposite sentiments towards each other. Perhaps this would be a worthwhile trade after all (and a larger, negative weight needs to be placed upon disfavor-favor relationships).
- (2) As mentioned in the previous point, Tommy dislikes Brandon. Trading Tommy in $P1$ for Jordan in $P2$ seems promising, But we have overlooked the fact that Colby and Jordan are our NA pair, and cannot be in the same patrol under any circumstances. Trading Colby for Jordan is another option, but Colby will be missed by Darwin, Evan, Cameron and Tommy (In other words, the whole of $P1$ minus Brandon).

On the whole, though, there are no obvious improvements that can be made to our algorithm’s partitions. We find that this is an assignment that could have reasonably been arrived at by Scoutmaster Bruce (granted the group dynamics have not changed since the scouts were surveyed). Furthermore, the solution took only 10 seconds to arrive at on a single core machine.

IMPROVEMENTS

While our algorithm demonstrably works well on smaller groups of 13 people, we do not expect our brute force solution to be tractable on larger groups wherein greater than 2 partitions are required. As a possible scenario, consider a company of 150 people to be divided into fixed-size teams of 10 individuals each. There are $150!/(10!)^{15}$ possible partitions, a number proportional to 10^{164} - meaning we

could instead use that computational time to enumerate the number of atoms in the universe... a tredecillion (10^{78}) times. Knowing this, perhaps it won't surprise the reader that this particular graph partitioning problem in general graphs is NP-complete - though there do exist approximation algorithms⁴. As scout enrollment increases, it becomes necessary to consider scenarios where we have both a large number of scouts in need of assignment and multiple patrols to choose from. Our algorithm could be extended to use brute force to find a globally optimal solution when it is deemed computationally feasible and to use an approximating algorithm to find a locally optimal solution otherwise.

CONCLUSIONS

Assigning individuals to compatible groups is hard (computationally speaking). When the group is small enough - as it is here - globally optimal solutions may be arrived at by checking all feasible solutions. But the amount of computation time needed to arrive at an answer grows exponentially with the number of individuals and groups. For example, finding an optimal partition of 13 scouts into *three* groups would take over 50 times as long as finding the optimal partition of 13 scouts into two groups (90,090 feasible combinations to check versus 1,716 in the two patrol case).

Nevertheless, our chosen objective functions seem to arrive at solutions that are able to hold up to human scrutiny, and perform well enough when total enumeration is possible. Larger problems may require approximating or clustering methods⁵.

ACKNOWLEDGMENTS

We would like to thank Scoutmaster Gene Bruce for his cordiality and for providing us the necessary data. We would also like to thank Professor Sara Billey for her mentorship and guidance provided throughout our project.

VERIFICATION STATEMENT

(Once we receive our verification statement, we will place it here.)

APPENDIX A

The other objective functions.

- (1) **AwkwardCut (+i)**. Same in all respects to MinCut, except for an additional penalty of i if there are two scouts, Scout A and Scout B, in the same patrol such that Scout A likes Scout B and Scout B dislikes Scout A.

$$AwkwardCut(I, J, i) := \left[\sum_{e=(k,j), k \in I, j \in J} c(e) \right] + i * |\{e_+ = (k, j), e_- = (j, k) : e_+, e_- \in I, c(e_+) = 1, c(e_-) = -1\}|$$

⁴B. W. Kernighan, S. Lin, *An Efficient Heuristic Procedure for Partitioning Graphs*. Bell System Technical Journal. **49** (1970), 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x

⁵Hartigan, J. A., and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm." Applied Statistics 28.1 (1979): 100.

- (2) **HybridCut.** A convex combination of MinCut and FriendCut, each with equal weight (that is, half the MinCut loss plus half the FriendCut loss).

$$HybridCut(I, J) := \frac{1}{2}MinCut(I, J) + \frac{1}{2}FriendCut(I, J)$$

APPENDIX B

The solution for finding the Stirling number of the second kind for $n = 13$ scouts to be arranged in $k = 2$ troops:

$$S(n, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^k \binom{2}{i} (k-i)^n$$

$$S(13, 2) = \frac{1}{2!} \sum_{i=0}^2 (-1)^2 \binom{2}{i} (2-i)^{13}$$

$$S(13, 2) = 4095$$

REFERENCES

1. B. W. Kernighan, S. Lin, *An Efficient Heuristic Procedure for Partitioning Graphs*. Bell System Technical Journal. **49** (1970), 291–307. doi: 10.1002/j.1538-7305.1970.tb01770.x
2. Desquelles, P. "Calculation of the Number of Partitions with Constraints on the Fragment Size." Phys. Rev. C Physical Review C 65.3 (2002)
3. Hartigan, J. A., and M. A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm." Applied Statistics 28.1 (1979): 100.