Big-Little Pairings

Kimberly Hsieh, Annabelle Huang, Jackie Pepe

21-393

December 14, 2014

Overview

Greek life is very prevalent at universities across the country. Each semester, new members join sororities and fraternities through formal recruitment. After recruitment, active members get matched with the new members and are considered "big siblings". The new members are considered the "little siblings". This matching problem has historically been done by the greedy algorithm. However, as membership size increases, the problem becomes larger, and the greedy algorithm is probably not the best way to match members. In addition, human factors may influence the matchings if the greedy algorithm is used. We believe that the assignment problem would be an improvement over the greedy algorithm. To model this, we are assuming that we have 5 new members (littles), and 10 active members (bigs). We also assume that each little will rank her preference of bigs from 1 to 3. In our model, we will assign scaled regrets for the cost of the assignments. Our aim is to minimize regret so we assign a 0 if the little gets her first preference, a 1 if the little gets her second preference, and a 2 if the little gets her third preference. If the little gets a big she did not prefer at all, we assign a regret value of 10 since we would never want this to happen. A sample of how the data was collected is below. Because of confidentiality reasons, we were not able to obtain the little's preferences from our sorority on campus. We ended up generating data to use by using a random number function in Excel.

	1st Pref	2nd Pref	3rd Pref
Little 1	Big 2	Big 4	Big 6
Little 2	Big 6	Big 8	Big 4
Little 3	Big 2	Big 4	Big 6
Little 4	Big 9	Big 6	Big 4
Little 5	Big 2	Big 4	Big 6

General Problem and Current Method

Currently, the sorority we are involved in, Alpha Phi International, uses the Greedy Algorithm to assign littles to bigs. Our New Member Educator is in charge of assigning the littles and bigs. She will go through the preference lists in order, starting with little 1. She will go through her preference list, and assign her to the big with the least regret (top choice available). She will then move on to little 2 and assign her to the next available big with the least regret. Finally, she will repeat for all the littles until she finishes assigning them all to bigs. This method is not only time consuming, but can also be biased at times, because the New Member Educator might make some human errors or might have biases towards or against certain members. We then looked into a specific formulation of integer programming - the assignment problem.

The assignment problem is a fundamental combinatorial optimization problem. There are a number of bigs and a number of littles. Any big can be assigned to a little, incurring some

regret that may vary depending on the big-little assignment. It is required that all littles be assigned exactly one big, but not all bigs have to be assigned a little. We want to minimize the total regret. This is the linear programming formulation for our model.

Variables:

$REGRET_{b,l}$ = scaled regret assigned according to number on preference list

B = set of all bigs

L = set of all littles

Decision Variables:

$A_{b,l} = 1$ if the little l is assigned to big b, 0 if the little l is not assigned to big b

Model:

$$\min REGRET = \sum_{b \in B} \sum_{l \in L} REGRET_{b,l} * A_{b,l}$$

subject to the constraints

$$\sum_{l \in L} A_{b,l} \leq 1 \ \forall b \in B$$

$$\sum_{b \in B} A_{b,l} = 1 \ \forall l \in L$$

$$A_{b,l} \in \{0, 1\} \ \forall b \in B, l \in L$$

In Microsoft Excel, our data for regret looks like this. Each column of data represents the preference list for that particular little and each row of data represents each big.

REGRET b/l	1	2	3	4	5
1	10	10	10	10	10
2	0	10	0	10	0
3	10	10	10	10	10
4	1	2	1	2	1
5	10	10	10	10	10
6	2	0	2	1	2
7	10	10	10	10	10
8	10	1	10	10	10
9	10	10	10	0	10
10	10	10	10	10	10

This is what our results look like in Microsoft Excel. The variables in the green matrix represent whether or not a little is assigned to a big. If we look down each column, there should only be one 1, and we can match that to the row number to see which big the little was assigned to. The row under the green matrix is the sum of all the entries in the respective column and it has to be equal to one, because every little is assured to get a big. The column on the right of the green matrix is the sum of all the entries in the respective row and it has to be less than or equal to one, because not all bigs will get littles. If it is a 1, then it means that the big got assigned a little; if it is a 0, then it means that the big was not assigned a little. On the far right in the blue box is the minimum regret for this set of data.

A b/l	1	2	3	4	5		
1	0	0	0	0	0	0 <=	1
2	1	0	0	0	0	1 <=	1
3	0	0	0	0	0	0 <=	1
4	0	0	1	0	0	1 <=	1
5	0	0	0	0	0	0 <=	1
6	0	0	0	0	1	1 <=	1
7	0	0	0	0	0	0 <=	1
8	0	1	0	0	0	1 <=	1
9	0	0	0	1	0	1 <=	1
10	0	0	0	0	0	0 <=	1
	1	1	1	1	1		
	=	=	=	=	=		
	1	1	1	1	1		

MIN regret
4

We used the same data for both the Greedy Algorithm and the Assignment Problem. Using the Greedy Algorithm gives us a total regret of 11, and one little has to be assigned a big who was not on her preference list, which is not ideal. We want to assign littles to bigs who are at least on their top three. The results from the assignment problem show that none of the littles were assigned to bigs that were not on their preference list, which is good, because we know that at least the littles will be assigned to someone who they like. The results improved by 64%, so we can conclude that the assignment problem formulation is a better way to assign big-little pairings compared to the greedy algorithm.

Greedy

Pairing	Regret
Little 1 and Big 2	+0
Little 2 and Big 6	+0
Little 3 and Big 4	+1
Little 4 and Big 9	+0
Little 5 and Big 10	+10

Total regret = 11

Assignment Problem

Pairing	Regret
Little 1 and Big 2	+0
Little 2 and Big 8	+1
Little 3 and Big 4	+1
Little 4 and Big 9	+0
Little 5 and Big 6	+2

Total regret = 4

IP improved the results by 64%

The General Stable Marriage Problem

Now that we have demonstrated a simple scenario where IP method improved the results by a significant amount, we decided to further explore and look for a general algorithm that we

can implement in our model with a larger and more complex data set.

Since we need to match two sets of members with given preferences, we first studied the stable marriage problem. This algorithm solves the following problem:

> Given n men and n women, where each person has ranked all members of the opposite sex with a unique number from 1 to n in order of preference, is it possible to marry the men and women together such that there are no two people of the opposite sex who would prefer each other over their current partners.

The most common algorithm known to solve the above problem is the Gale-Shapley algorithm, developed by David Gale and Lloyd Shapley in 1962. The algorithm is iterative with details of the iterations listed below.

Gale-Shapley Algorithm Iterations:

1. Each unengaged man proposes to the woman ranked highest on his preference list and to whom he has not already proposed.

2. Each woman who receives a proposal becomes engaged to the man ranked highest on her preference list out of all of her suitors.

    i. If a woman was already engaged and receives a proposal from a man who ranks higher on her preference list than her current fianc, she can dissolve her current engagement and enter into a new one. Her previous fianc is then considered unengaged and can continue with his proposals.

3. Once the matchings have been made, the process reiterates with the remaining unengaged men continuing through their list of proposals.

The Gale-Shapley algorithm is considered effective because it guarantees two very impor-

tant details: 1) every man and woman eventually ends up engaged and 2) at the end, every engagement is stable. That is,there are no two people who would consider breaking their current engagements to be with each other. However, this does not necessarily mean that each person is matched with their top preference.

Now we show an example of solving the stable marriage problem using the GS algorithm with n = 3 (i.e. 3 Bigs and 3 Littles). Their preferences are listed below:

	Big 1	**Big 2**	**Big 3**		**Little 1**	**Little 2**	**Little 3**
1st Pref	L3	L3	L2		B3	B1	B1
2nd Pref	L1	L2	L1		B2	B3	B2
3rd Pref	L2	L1	L3		B1	B2	B3

Iteration 1:

-L1 proposes to B3; B3 accepts and they become engaged ⇒ (L1,B3)

-Both L2 and L3 proposed to B1; B1 prefers L3 over L2 and thus accepts L3?s proposal ⇒ (L3,B1)

-L2 remains unengaged; B2 received no proposals and therefore also remains unengaged

Iteration 2:

-L2 proposes to next on list, B3; C is engaged to L1, but prefers L2 over L1 so breaks up with L1 and becomes engaged to L2 ⇒ (L2,B3)

-B1 receives no more proposals and therefore remains with L3

-Now L1 is unengaged, as is B2

Iteration 3:

-L1 proposes to next on list, B2; B2 accepts ⇒ (L1,B2)

-There are no more unengaged males and thus the algorithm is complete.

{(L1,B2), (L2,B3), (L3,B1)} is the resulting set of stable matchings. No Big/Little pairs has an incentive to break up, verifying the fact that the Gale-Shapley algorithm is effective. We can see that the stable marriage algorithm can be related to Greek life big/little pairing as mentioned in the Introduction, the process is dependent upon mutual selection and the matching of preferences. However, there are important discrepancies we noted, including the fact that the number of Bigs and potential Littles are not the same, meaning that the problem is no longer mimics the matching up of n men and n women. In addition, since each Little only provides her top-k preferences instead of ranking all n bigs, we have a partial, incomplete preferences list.

The Gale-Shapley algorithm is ideal only for the stable marriage problem and its use in this application may not provide optimal results. An algorithm that solves a variation of the problem that accounts for these mentioned modifications should be considered instead.

Partial Preferences

A variety of extensions of the basic problem have been studied. In the Stable Marriage problem with Incomplete lists (smi), the numbers of men and women need not be the same, and each person's preference list consists of a subset of the members of the opposite sex in strict order. A pair (m, w) is acceptable if each member of the pair appears on the preference list of the other. A matching M is now a set of acceptable pairs such that each person belongs to at most one pair. In this context, (m, w) is a blocking pair for a matching M if (a) (m, w)

is an acceptable pair, (b) m is either unmatched or prefers w to M(m), and likewise (c) w is either unmatched or prefers m to M(w). As in the classical case, there is always at least one stable matching for an instance of smi, and it is straightforward to extend the Gale / Shapley algorithm to give a linear-time algorithm for this case. In particular, Gale and Sotomayor studied the extension where each person is allowed to declare one or more unacceptable partners, meaning each person?s preference list may be incomplete. They showed that every stable matching for a given smi instance has the same size and matches exactly the same set of people, and that there is a polynomial-time algorithm which determines whether there exists a stable matching and finds one if one exists. Thus the problem does not become essentially harder.

Conclusion

In conclusion, it appears that the results have improved. More littles are getting bigs that they actually preferenced. The assignment problem is also easy to implement and very systematic. It helps the new member educator by making the problem a lot less time consuming. New member educator ran the program on the data from fall 2014 and claimed that results improved. Finally, this model has prevented biased matchings from happening since it is done by the computer, and human error will not come into play.

Further Discussion

The assignment problem has many applications in the real world, such as course scheduling and selection. The preferences would include seniority and major. A similar problem would be dorm selection for incoming freshmen. A third application this problem applies to could be plain, train, or bus seating assignment. Most boarding occurs in some sort of predetermined order, whether it be by time of arrival or a special class that a traveller falls in. Based on the type of preference used for the system, travelers will be assigned to their seat in a particular

way.

Sources

Gale, David, and Marilda Sotomayor. "Ms. Machiavelli and the Stable Matching Problem." The American Mathematical Monthly, 1 Apr. 1985. Web. <http://pareto.uab.es/jmasso/pdf/GaleSotomayorAMM1985.pdf>.

Gelain, Micro, Maria Silvia, Francesca Rossi, Brent Venable, and Toby Walsh. "Male Optimal and Unique Stable Marriages with Partially Ordered Preferences." University Di Padova, Italy. Web. <http://www.math.unipd.it/ frossi/stc-care5-cr.pdf>.

Irving, Robert, David Manlove, and Gregg O'Malley. "Stable Marriage with Ties and Bounded Length Preference Lists." University of Glasgow. Web. <http://pdf.aminer.org/001/277/519/stable_marriage_with_ties_and_bounded_length_preference_lists.pdf>.

Iwama, Kazuo, and Shuichi Miyazaki. "A Survey of the Stable Marriage Problem and Its Variants." International Conference on Informatics Education and Research for Knowledge-Circulating Society. Web. <http://148.204.64.201/paginas anexas/ALGORITMICA/papers/preg1/scfu_21_paper.pdf>.