The Sorting Hat Goes to College

Andrew Beveridge and Stan Wagon, Macalester College
abeverid@macalester.edu
wagon@macalester.edu

1. Introduction

In the Harry Potter stories [R], each new year at the Hogwarts School for Witchcraft and Wizardry starts with the ceremonial assignment of the new first-year students to one of the four houses: Griffindor, Hufflepuff, Ravenclaw, and Slytherin. This is a milestone for the young students because their assigned houses greatly influence their future direction. The crucial assignment process is entrusted to the magical Sorting Hat, which provides a moment of revelation for the student. Indeed, this sentient garment uses its magical perception to reveal the student's personality and strengths, and consequently places the student into the best-suited house. However, we cannot help but observe another pattern in the assignments. Every year, the 40 students are split evenly, both among houses and by gender. Perhaps the Hat respects some constraints as it divides the students.

This constrained optimization problem captures a fanciful moment of high drama. However, a strikingly similar process takes place annually on college campuses across the United States. Rather than assigning students to houses, colleges must distribute incoming students among an offering of first-year seminars. These courses ease the transition into college by introducing the paradigms, expectations, and standards of higher education [B]. Like the houses of Hogwarts, many first-year seminars strive to develop community within the classroom, the student body, and the institution as a whole. Some colleges use seminars with uniform content, but many offer a variety of discipline-specific seminars whose content is as disparate as the interests of the faculty. In this latter setting, assigning first-years to seminars becomes remarkably similar to the job of the Sorting Hat.

Macalester College, a liberal arts college with about 2000 students, offers discipline-specific first-year courses. Macalester places great institutional effort in developing these courses and supporting their efficacy. Likewise, matching each student to a desired course is also taken very seriously. Prior to 2009, this assignment was performed by hand using a labor-intensive process. This process was ripe for improvement, both in terms of efficiency and quality of assignment. Without access to sorcery, we decided to use the next best thing: mathematics! This paper describes the evolution of our automated procedure for creating an assignment.

Our first solution leveraged the classic Minimum Weight Perfect Matching algorithm from graph theory, with excellent results. More recently, we set out to improve our process by developing a more flexible approach using Integer Linear Programming. This paper traces the evolution of this process and discusses the resulting measurable improvements.

2. First Year Course Assignment at Macalester

Every department at Macalester offers one or more first-year seminars, called First-Year Courses (FYCs). In a given year, no two FYCs are alike. During early summer, students are given course descriptions and rank their top four choices. The Academic Programs office (AP) compiles the student preferences and is responsible for assigning students to courses, while respecting some modest constraints on the enrollment profile of each seminar. Typically, about 500 incoming students must be placed into about 35 FYCs. The goal is to place each student into a course that he or she ranked, while also trying to maximize overall student satisfaction. Roughly speaking, this means trying to have many first- and second-choice placements, and far fewer third- and fourth-choice placements.

Academic Programs also tries to enforce the following constraints, designed to keep a uniform experience among classes. First, the class size must be between 10 and 16 students. Second, the demographics of each class should be roughly comparable to the entire student body. Historically, this has meant enforcing a gender balance by reserving 4 seats for men and 4 seats for women. Macalester's student body is 60% female, so this constraint becomes relevant in a few seminars each year. Finally, international students make up 11% of the student body, and AP has found it useful to cap the number of international students in any course at 8.

In a given year, it may not be feasible to assign each student to a ranked choice while enforcing all constraints. In these situations, AP allows some isolated constraint violations so that every student ends up in a desired course; the alternative — placing a student in an unranked course — is deemed unacceptable.

Prior to 2009, the assignment was performed by hand. Each student's four preferences were listed on a sheet of paper and these were organized into piles, each pile representing a seminar. Two staff members would spend a week looking for beneficial chains of swaps that would improve the overall profile of the assignment. Ultimately, this method worked well: Figure 1 shows some historical results for these manual assignments. In §4 we will show how to get an assignment for the 2004 data that puts 97.3% into their top two choices (as opposed to the 87.4% from Fig. 1). On average, AP placed 87% of students into their first or second choices.

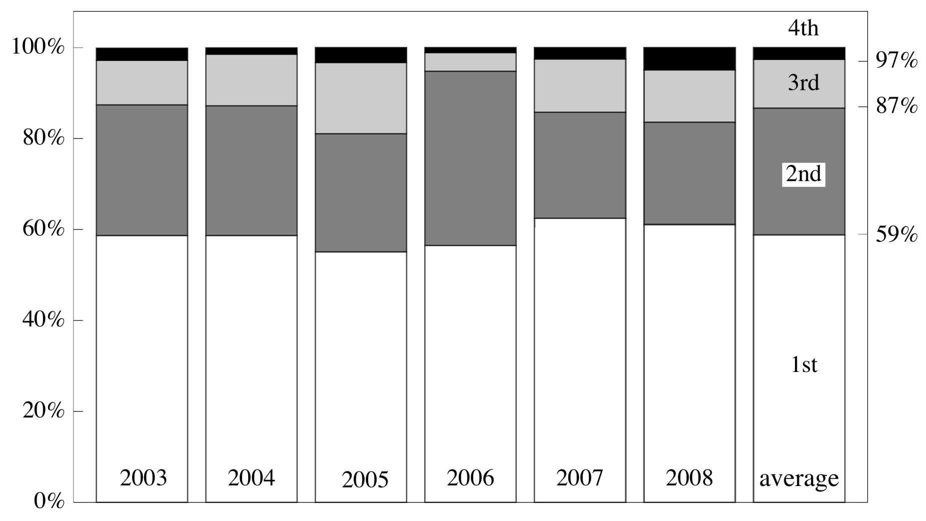

Figure 1: Manual assignment profiles, 2003–2008

3. The Hungarian Sorting Hat

In 2008, the first author and Macalester senior Sean Cooke tackled the FYC Assignment Problem in a senior seminar in Combinatorial Optimization. Our goal was to automate the process by converting the problem to an Assignment Problem (described below). In addition to the benefits of automation, we were confident that we could also improve the quality of the assignment. We approached this as a consulting project, where the “client” was the AP office. Through a series of meetings with them, we identified their requirements, constraints, and priorities. We used historical data to fine-tune our solution, presented prototypes, and received feedback on their performance. This sustained dialog was crucial in the final design, and also to successfully “sell the client” on adopting the algorithm. In particular, we showed AP that our algorithm, dubbed the Hungarian Sorting Hat because of its use of the Hungarian matching algorithm, outperformed their manual assignments on the historical data. They adopted our automated framework in summer 2009, and have used it ever since, but with an algorithm change in 2013 (§4).

Figure 2 shows the performance of the Hungarian Hat from 2009 to 2012. The average (1st, 2nd, 3rd, 4th) percentages for the manual assignments of Figure 1 were 58.8%, 27.9%, 10.7%, 2.6%. The average performance of the Hungarian Hat is 59.5%, 31.3%, 8.3%, 0.9%. Even though the years are different than those in Figure 1, this improvement is typical for the comparison: the Hungarian Hat placed 4.1% more students in their first or second choices, compared with the manual assignments, as well as many fewer into a fourth choice.

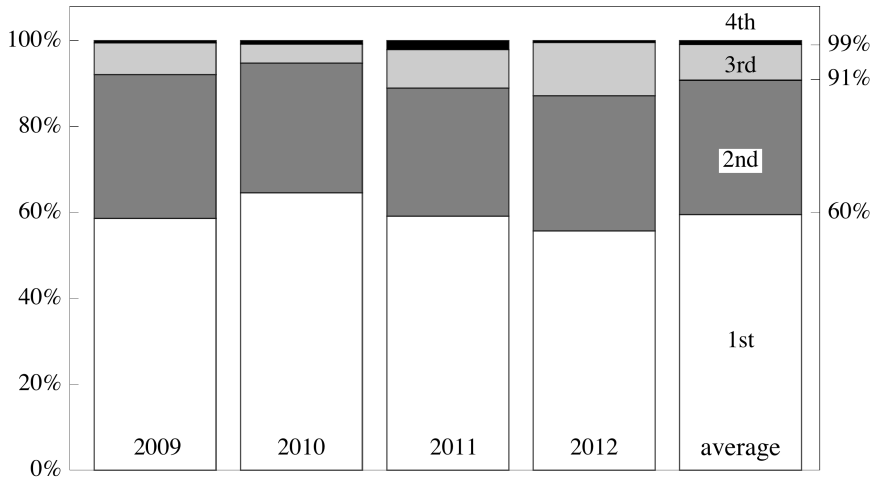

Figure 2: Assignments by the Hungarian Sorting Hat, 2009–2012.

We now describe the mathematics behind the Hungarian Sorting Hat. The FYC Assignment Problem is a variant of the classic Assignment Problem (see [C]). Suppose that we have *n* workers and *n* jobs to be filled. Each worker is qualified to perform a subset of the jobs, though his salary depends upon the job assigned. The objective is to find a matching of workers to jobs that minimizes total cost. This situation can be modeled by a weighted bipartite graph, where the *n* vertices in one part correspond to the workers and the *n* vertices in the other part represent the jobs. If worker *i* is qualified to perform job *j* at the cost c_{ij}, then we connect the corresponding vertices with an edge of that weight. This optimization problem is solved by the very fast Hungarian Algorithm, developed by Kuhn, who was extending the work of the Hungarian mathematicians Kőnig and Egerváry. A detailed description of the method is in [BM, chap 5]. The algorithm analyzes the weighted bipartite graph, and returns a minimum weight perfect matching. This matching corresponds to an optimal assignment of workers to jobs. The basis of the algorithm is the use of swaps along augmenting paths, a very natural idea and one that was used in an informal way by the AP staff when the job was handled manually.

Our main challenge was to convert the FYC Assignment Problem into a classic Assignment Problem. This involved two steps. First we created a bipartite graph that enforced the college constraints on class size and class demographics. Second, we chose a weighting scheme that ensured that the Hungarian Algorithm made trade-offs that are consistent with the goals and priorities of the AP office. As we have already seen in Figure 2, the resulting matchings were of high quality during these years.

We summarize the features of the resulting graph. For a more thorough explanation, see [BC]. In order to run the Hungarian Algorithm, we need a bipartite graph *G* with parts *X* and *Y*, where $|X| = |Y|$, so we start by describing these sets. The set *X* will represent the students (workers) and the set *Y* will represent the classes (jobs). For each class, we place 16 vertices into *Y*, one for each seat available. Next, we place a vertex in *X* for each student. Of course, there are more class seats than students, so we add some additional “dummy” vertices to *X* to balance this out. These dummy vertices will ultimately correspond to empty class seats.

Next, we describe the edge structure of G from the seminar perspective; a small part of the graph is in Figure 2. This edge structure enforces class size and demographic constraints. A given seminar has 16 corresponding vertices. Eight of these vertices are adjacent to every student interested in the course. Four of the vertices are adjacent only to interested US women, and the final 4 vertices are adjacent exclusively to interested US men. A fully enrolled class must have at least 4 US women and 4 US men, and a maximum of 8 international students. Finally, each dummy vertex is adjacent to one US woman's seat, one US man's seat, and four generic seats. This means that there can be at most 6 unfilled seats, for a minimum enrollment of 10 students. Furthermore, a class of size 10 is only required to have 3 US women and 3 US men.

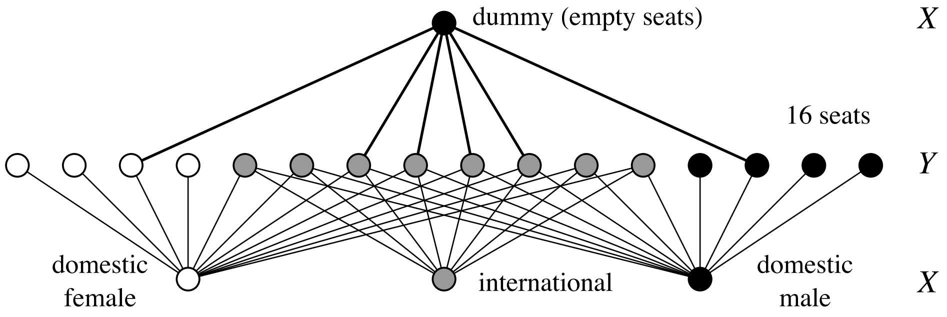

Figure 3: The edges connecting the 16 seats in one class to the four types of student: domestic female, international, domestic male, and dummy students. The edge structure assures that a perfect matching will respect the desired constraints.

Finally, we describe the edge weights. We ultimately settled on a geometric weighting scheme, where the weights of first, second, third, and fourth choice classes were given by the weight vector $(1, \alpha, \alpha^2, \alpha^3)$ for a well chosen $\alpha > 1$. This scheme has a natural interpretation in terms of trade-offs one is willing to make in swapping students between classes. The base weight α is the upper bound on the number of students k we are willing to move down from first choice to second choice in order to move a single person up from a third choice to second. This leads to a smaller weight matching precisely when $0 > k(\alpha - 1) + (\alpha - \alpha^2) = (k - \alpha)(\alpha - 1)$, a condition equivalent to $k < \alpha$. Other swaps have analogous interpretations. In practice, AP would often perform 2-to-1 and 3-to-1 swaps of this kind. Therefore, we ended up using $\alpha = 3.5$ as our geometric weight (which also provided results that were stable under minor changes to the weight value).

The last weight we must describe is the one used for the dummy students. The algorithm should always prefer to place a real student in a class before placing a dummy student. This can be accomplished by giving the dummy edge a weight much larger than any of the "real" edges in the graph. In practice, the value α^4 does the trick. And of course, dummy students have no class preferences, so every dummy edge in the graph is given this same weight. This concludes the description of the weighted graph model.

This model was a useful tool for attaining a desirable outcome: when there is an assignment that meets all constraints, the algorithm is guaranteed to find it, and it is very fast. But it can and does happen that the graph does not have a perfect matching (a matching that covers all vertices). This will happen if the constraints cannot all be met: perhaps there is a course that only 3 men have chosen as possibilities; or 3 courses that have male preferences with overlap so that not all of them can have 4 men. In such cases there is no perfect matching in the graph and the algorithm fails. Another difficulty (which arose for us in 2013) is when student choices are such that the class-size constraint (either the minimum or the maximum) cannot be met. For example, the data might force there to be, say, 12 classes that must increase enrollment to 17. Some of these problems can be handled by setting up a new graph but that is tedious, and worse, there is no global solution. For example, one cannot set up a graph that will provide at most 12 classes of size 17. Instead, one must specify which particular classes will rise to 17.

Yet another problem arises when a college has a different set of constraints. For example, Beloit College uses our methods, but wishes to avoid having students from the same high school placed in the same class, something that cannot easily be set up in the graph environment. We imagine that some colleges might want to restrict on other demographics, such as state of residence or athletic participation. Such issues (infeasible or complicated constraints) will lead to trouble for the Hungarian Sorting Hat. This led us to the Integer Linear Programming method, which is easy to set up, can handle a wide variety of constraints, and will always perform at least as well as the graph theory method.

4. A Flexible Sorting Hat

Classic linear programming (LP) problems can be solved very quickly by the venerable simplex method. This is quite similar to Gaussian elimination: a straightforward algebraic process. For combinatorial optimization one often needs integer linear programming (ILP), where the variables are constrained to have integer values. Algorithms for ILP work by making repeated (100s or 1000s) calls to LP, using a branch-and-bound approach. A very brief introduction to branch-and-bound is in [W]; see [T] for more detail.

In LP or ILP we need a linear system of inequalities and an objective function so that (a) a solution to the system gives as a workable assignment of students, and (b) minimizing the objective gives an assignment that maximizes student happiness.

The basic variables are easy to set up. Because we will use ILP, all variables are restricted to be integers. Let the students be S_i, $1 \le i \le n$, and the courses C_j, $1 \le j \le m$. Let the preferences of S_i be P_i, a set of four integers, the indices for courses the student is willing to take. For each course C_j, let X_j be the set of student indices i so that $j \in P_i$; that is, X_j is the set of students having C_j as one of their choices; subdivide this farther into X_j^{male} and X_j^{female} to denote the male or female students willing to take C_j, and also let X_j^{intl} be the set of international students willing to take the course. As noted, student preferences can be problematic; an extreme case would be some $X_j = \emptyset$, but if some X_j is very small that will also require some manual intervention, or possibly the canceling of a class. We will describe the setup for the constraints as given above; a reader facing different constraints should be able to modify the setup without difficulty.

For each pair (S_i, C_j) where course $j \in P_i$, introduce a variable $x_{i,j}$, which is to be 0 or 1, with a 1 indicating that S_i is assigned to C_j. This contributes $4\,n$ variables.

The Basic Constraint. For each variable, $0 \leq x_{i,j} \leq 1$. Also, each student gets exactly one course: for each $i \leq n$, $\sum_{j \in P_i} x_{i,j} = 1$.

The Objective Function. We want to minimize unhappiness. So if $P_i = \{j_{i,1}, j_{i,2}, j_{i,3}, j_{i,4}\}$, the objective is to minimize $\sum_{i=1}^{n} x_{i,j_{i,1}} + \alpha\, x_{i,j_{i,2}} + \alpha^2\, x_{i,j_{i,3}} + \alpha^3\, x_{i,j_{i,4}}$, where $\alpha > 1$ is a fixed parameter as described in §3.

Course Size Constraint. There will be an upper bound, U, and a lower bound, L, on course size; for each $j \leq m$, $L \leq \sum_{i \in X_j} x_{i,j} \leq U$.

Gender Constraint. A typical gender constraint might be that each course contains at least four students of each gender: for each $j \leq m$, $\sum_{i \in X_j^{\text{male}}} x_{i,j} \geq 4$ and $\sum_{i \in X_j^{\text{female}}} x_{i,j} \geq 4$.

International Constraint. Each course contains at most B international students: for each $j \leq m$, $\sum_{i \in X_j^{\text{intl}}} x_{i,j} \leq B$.

The preceding are the ideal constraints, but it can happen that not all of them can be met. One way to handle typical problems is to introduce extra variables. For ease of exposition, assume $(L, U) = (12, 16)$, but other values can be handled in the same way. Since allowing a course to grow to 17 is common, as is allowing a course to have fewer than 12, let $q_{j,s}$ be a 0-1 variable that indicates that the size of C_j is s, where $9 \leq s \leq 17$. Assume that four parameters Q_{17}, Q_{11},Q_{10}, Q_9 are given; they represent the maximum number of classes whose size is allowed to be 17, 11, 10, 9, respectively. Now the number of variables is $4\,n + 4\,m$.

Then the ideal size constraint is replaced by the following:

for each j and s: $0 \leq q_{j,s} \leq 1$;
$\sum_j q_{j,17} \leq Q_{17}$ (number of courses with 17 students is at most Q_{17});
$\sum_j q_{j,11} \leq Q_{11}$ (number of courses with 11 students is at most Q_{11});
$\sum_j q_{j,10} \leq Q_{10}$ (number of courses with 10 students is at most Q_{10});
$\sum_j q_{j,9} \leq Q_9$ (number of courses with 9 students is at most Q_9);

and for each $j \leq m$:

$\sum_{s=9}^{17} q_{j,s} = 1$ (size of C_j is a unique value between 9 and 17);

$\sum_{i\in X_j} x_{i,j} = \sum_{s=9}^{17} s\, q_{j,s}$ (size of C_j is given by the q-indicator).

A similar enhancement relaxes a gender requirement: replace the ideal male constraint by the addition of indicator variables $y_{m,j}$ so that a value of 1 means that C_j has m males. The new constraints are $\sum_{m=0}^{17} y_{m,j} = 1$ and $\sum_{m=0}^{17} m\, y_{m,j} = \sum_{i\in X_j^{\text{male}}} x_{i,j}$ (both for all j), and, say, $\sum_j y_{0,j} = \sum_j y_{1,j} = \sum_j y_{2,j} = 0$ and $\sum_j y_{3,j} \leq Y_3$, where Y_3 is a new parameter that bounds the number of courses that can be male-deficient by 1. And can also bring in Y_2 if needed for courses with two males. The variable count is now $4\,n + 22\,m$.

Mathematica has ILP available and can handle a typical setup (there are about 3000 varibles in our case) in under five seconds, returning either the optimal solution or a message indicating that the constraints are not feasible. This is so fast that one can run the routine repeatedly to learn what is feasible and what is not. It is when the basic setup is infeasible that ILP is most useful, since the auxiliary bounds (Q_i, Y_3) can be tweaked until a solution is found.

Before using ILP for real, we tried it on past data sets. The 2004 case was interesting: The profile — 1st through 4th percentages of the manual assignment — was (59 %, 29 %, 11 %, 1.4 %). The data were such that there had to be a course with only 2 men; other than that the male constraint could be satisfied. Because that course was easily identified, the graph matching approach would work since we can just adjust the edge set for that one course. But things can be more subtle, involving several courses, and there might not be a simple edge-adjustment approach available.

Using ILP we can vary the settings for the number of male-deficient courses (the Y_3 and Y_2 mentioned earlier), which is how we learn the minimum violations. We can also add constraints of the form "number of 4ths is at most P_4" as a way of discovering the minimum number of 4th choices (it is 6), and similarly for 3rd choices (it is 8). Using $\alpha = 1.5$ gives the profile (69 %, 27 %, 3.3 %, 1.2 %), while using $\alpha = 5.5$ gives (65 %, 33 %, 1.5 %, 1.2 %). The latter has fewer in #3 and #4, but the former has quite a few more in #1. This sort of trade-off is what we typically present to AP. Both are significantly better than the manual result. As a bonus, we learn that the $\alpha = 5.5$ profile is completely optimal as far as minimizing #3s and #4s. This leads to another optimization idea, which is to first minimize the number of 4th choices, then minimize the number of 3rd choices subject to the number of 4th choices being minimal, and then once more for 2nd choices. Such a method is useful in some ILP problems, but we found that here it is equivalent to using a large value of α.

Here is how things worked in July 2013 at Macalester. There were 550 students and 34 courses. But the data set was more difficult than average and there could be no assignment that respected either the upper bound of 16 or the minimum-male requirement of 4. It was easy to learn that there had to be at least twelve courses of size 17 and at least two courses with only 3 males. Because the choice of twelve courses to expand is not clear, there is no easy adjustment to the graph method that can handle this. But ILP has no problem, setting the oversize parameter Q_{17} to 12.

Figure 4 shows a chart of the sort we presented to AP in 2013. It shows that: twelve courses have size increased to 17 students, no course has fewer than 14 students, and two courses have only 3 males. The preference profile is: 55% into choice #1, 82% into choice #1 or #2, and 3.8% into choice #4. This is definitely worse than other years, but this data set was less than ideal.

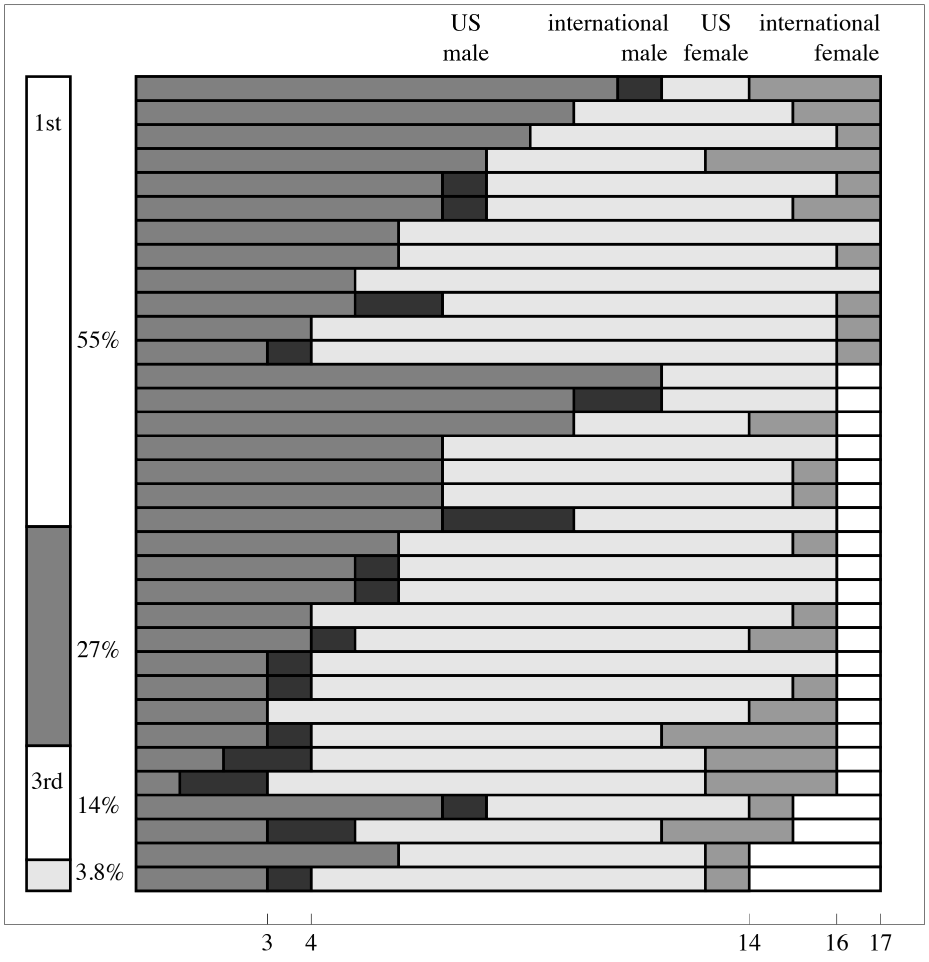

Figure 4: The profile for an assignment produced by ILP for actual 2013 data. Two courses have three males, and twelve courses have 17 students.

In 2013, our ILP method was also used at Beloit College, with 96% getting choice #1 or #2, and 2% getting choice #4. One word of warning though: ILP is an NP-complete problem and so if the class size grows into the thousands it will likely be too slow.

5. Conclusion

This topic is useful to colleges in several ways. It helps the mathematics department, since in an optimization or applied mathematics course, the students find the whole discussion fascinating, as it involves something they are very familiar with. They quickly appreciate the potential of both graph theory and ILP to be useful in diverse situations (an especially noteworthy application of both ILP and matching algorithms is in the optimal arrangement of kidney transplants [A]). And of course the methods are directly beneficial to the college administration. It is clear that a sophisticated algorithm will outperform any manual approach to the FYC assignment problem.

Any reader interested in learning how our approach could work at their institution should contact us.

References

[A] D. J. Abraham, A. Blum, T. Sandholm, Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges, *Proc. 8th ACM Conf. on Electronic Commerce* (2007) 295–303. Available at http://www.cs.cmu.edu/afs/cs/Web/People/sandholm/kidneyExchange.EC07.withGrantInfo.pdf

[B] D. Berrett, The many faces of the freshman seminar, *Chronicle of Higher Education*, July 29, 2013. Available at http://chronicle.com/article/The-Many-Faces-of-the-Freshman/140543

[BC] A. Beveridge and S. Cooke, The mathematical Sorting Hat, *The UMAP Journal of Undergraduate Mathematics and Its Applications*, 33:2 (2012) 99–117. Available at https://www.macalester.edu/~abeverid/papers/sortinghat.pdf

[BM] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, Amsterdam, Elsevier, 1976.

[C] W. Cook, W. Cunningham, W. Pulleyblank and A. Schrijver, *Combinatorial Optimization*, New York: Wiley, 1998.

[R] J. K. Rowling, *Harry Potter and the Sorcerer's Stone*, New York: Scholastic, 1998.

[T] P. R. Thie, *An introduction to Linear Programming and Game Theory,* 2nd ed., New York, Wiley, 1988.

[W] S. Wagon, An algebraic approach to geometrical optimization, *Math Horizons* (Feb. 2012) 22–27.