The Mathematical Sorting Hat: First-Year Course Assignment at Macalester College

Andrew Beveridge and Sean Cooke
Department of Mathematics, Statistics and Computer Science
Macalester College
Saint Paul, MN 55105

1 Introduction

The most well-known matching problem in recent popular culture comes from J. K. Rowling's *Harry Potter* series [Rowling, 1998]. Each year, roughly 40 new first-year students at Hogwarts School of Witchcraft and Wizardry are assigned to one of the residential Houses (Griffindor, Ravenclaw, Hufflepuff and Slytherin). This duty is performed by the magical Sorting Hat, which considers each student in turn and determines the best match for that student among the four Houses. The Sorting Hat uses its magical perception to assign students to a house that best matches their personality. However, we also notice that the assignments split fairly evenly, with each House receiving about 5 girls and 5 boys. Perhaps this house distribution and gender balance reflects a secondary goal of the Sorting Hat.

A similar sorting occurs annually on college campuses across the United States. The past two decades have witnessed the rise of the *first-year seminar*, a curricular strategy to help students transition from high school to college [StateUniversity.com, 2011]. A seminar is a small course that emphasizes active learning, discussion and exchange of ideas. First-year seminars leverage the closeness of this format to introduce the paradigms and standards of higher education. Additionally, first-year seminars at some institutions also strive to develop community within the classroom, the student body, and the institution as a whole. Many colleges devote considerable effort in assigning students to these seminars, trying to create an immediate sense of belonging, just as the fictional Sorting Hat achieves at Hogwarts.

This paper describes the development and implementation of an automated procedure for assigning first-year students to seminars at Macalester College. This solution, which we call the

Mathematical Sorting Hat, was adopted by Macalester College in 2009. It is used annually to match incoming students to seminars. The Hat models this matching problem using a weighted bipartite graph that captures both the student preferences and college constraints. We run the well-known Hungarian algorithm to find a minimum weight perfect matching of this graph. This matching corresponds to an optimal assignment of students to seminars.

When run on historical data, the Hat finds student assignments that are quantitatively better than the manual assignments used in those years. In the three years since adoption, the Hat has produced high quality matchings of students to seminars. The algorithm, implemented in Matlab, matches roughly 500 students to seminars in about 70 seconds on a 2.8 GHz Mac Pro. It saves 40 person-hours of labor. It also provides repeatability and customization. The goal of this paper is to explain the modeling process which allowed us to convert this applied problem into a classical optimization problem. Many colleges are faced with similar assignment problems, and we hope that this paper inspires others to consider automated methods.

This paper is organized as follows. In Section 2, we quickly survey matching problems and their recent applications. We then give a historical discussion the First-Year Course Assignment Problem (FYCAP) at Macalester College. In Section 3, we develop our weighted bipartite graph model for FYCAP. Section 4 reflects on our solution, and discusses some possible generalizations. Finally, Appendix A collects some implementation notes concerning repeatability, missing data and fairness. The code for the Mathematical Sorting Hat is available on the COMAP website.

2 Background

2.1 Matching Algorithms and their Applications

Before discussing our matching problem, we describe some recent applications of matching problems and their variants. Among the most prominent examples is the National Resident Matching Program (NRMP) [National Resident Matching Program, 2011], which places $30,000$ graduating medical students into hospital residencies each year. This is an example of a *stable marriage problem.* The preferences of both applicants and hosting institutions are taken into account. The algorithm finds a matching that is *stable* in the sense that no two matched pairs would unanimously agree to switch assignments. Since 1998, the NRMP uses an applicant-proposing deferred-acceptance algorithm ([Gale and Shapley, 1962] and [Roth and Peranson, 1999]) to find this stable matching. This procedure gives applicant preferences a slight edge over institutional preferences. Indeed, there is an inherent asymmetry to this problem, since the parties doing the "proposing" would never do better (and may do worse) if the roles were reversed.

Stabling matchings have also been used to improve the annual assignment of $90,000$ teens to public high schools in New York City [Abdulkadiroğlu et al., 2005]. This implementation employs the same stable matching strategy used by the NRMP. However, the algorithm has been adapted to co-exist with previous mechanisms, such as lotteries and separate offers from specialized schools. These alterations do lead to some mathematically unstable matchings, which must be tolerated in a complex system with legacy procedures.

Dorm room allocation is another problem relevant to campus life. In the standard formulation, students may choose to remain in their current room, or enter the applicant pool. Each applicant is randomly assigned a priority (perhaps weighted by seniority). In the simplest algorithm, applicants choose an available dorm room according to the priority ordering. However, this algorithm leads to an inefficiency: many students will opt to remain in their current room, for fear of ending up in a worse option. A simple and theoretically superior extension was proposed in [Abdulkadiroğlu and Sönmez, 1999]. The priority queue is created as before. Now, an applicant can request an occupied dorm room, at which point the current occupant moves to the top of the queue, directly in front of the requester. The current occupant can choose a new room (possibly requesting another occupied room, which pushes that occupant to the top of the queue), or decide to remain in his current one.

Recently, Abraham, Blum and Sandholm [Abraham et al., 2007] designed a program to improve efficiency in matching donors to recipients in kidney transplants. Excepting close family members, finding a compatible match for a potential recipient is very difficult. A recipient may have a friend who is a willing donor, but who has an incompatible tissue type. In this case, the recipient-donor pair attempt to find another recipient-donor pair for which a cross-transplant would be a better match. The algorithm also tries to connect larger chains of people, though the logistics of executing multiple simultaneous transplants becomes a limiting factor. The first cross-transplant using a match from this algorithm was performed in December 2010 [Emspak, 2010].

Finally, we mention that the Sorting Hat of Hogwarts appears to be performing an *on-line weighted bipartite matching algorithm.* In an on-line algorithm, we must process the information and make decisions as we go, rather than considering the whole input before making any assignments. An on-line algorithm is forced to make decisions that ultimately may not be globally optimal. Indeed, it has been shown [Khuller et al., 1991] that no on-line weighted bipartite matching algorithm can achieve the performance of an optimal off-line algorithm (which has access to all of the data) for all inputs. However, this result should not affect our faith in the Sorting Hat: a magically sentient hat is probably not bound by the same rules as a computer algorithm.

2.2 The Assignment Problem

The *Assignment Problem* (cf. [Cook et al., 1998]) is the matching problem that is best suited for our seminar assignments. Suppose that we have n workers and n jobs to be filled. Each worker is qualified for a subset of jobs, and his salary varies depending on the job assigned (he may have a higher certification for one job versus another). The objective is to find a matching of workers to jobs that minimizes total cost.

We recall some standard terminology from graph theory (cf. [West, 2000]). A *graph* $G = (V, E)$ consists of a set of vertices V along with a set of edges E consisting of pairs of vertices. When $v_1, v_2 \in V$ and $e = (v_1, v_2) \in E$), we say that v_1 and v_2 are *adjacent* and that edge e is *incident* with both v_1 and v_2. A *matching* M of G is a set of edges such that each vertex is incident with at most one edge. When a vertex v is incident with an edge in M, then we say that M *covers* v. A *maximum matching* is one of maximum size and a *perfect matching* is one that covers all vertices. A perfect matching is also a maximum matching, but the reverse statement is not true (for example, consider a graph on an odd number of vertices).

In the Assignment Problem, we create a graph with one vertex per worker and one vertex per job. We add an edge between worker x and job y whenever x is qualified to perform job y. This results in a *bipartite graph* whose vertex set can be partitioned into disjoint sets X, Y so that every edge has one end in X and the other in Y. Finally, we address the cost of a matching. Our workers earn different salaries, depending on their job. We model the cost using a *weighted graph*, where each edge $e \in E$ is assigned a weight w_e, corresponding to the cost of that assignment. Our objective is to find a perfect matching of minimum weight.

This optimization problem can be solved by the Hungarian algorithm, which was developed by Kuhn by extending the work of the Hungarian mathematicians Kőnig and Egerváry (cf. [Cook et al., 1998, Evans and Minieka, 1992]). The fundamental strategy of this algorithm (as well as many other matching algorithms) is to search for augmenting paths. Suppose that you start with a matching M on a graph G. An *M-alternating path* is a set of successive edges of G which are alternately in M and not in M. An *M-augmenting path* is an M-alternating path that begins and ends at uncovered vertices. Such an augmenting path identifies a series of swaps that results in a larger matching. Essentially, the Hungarian algorithm considers a series of subgraphs $G_1 \subset G_2 \subset G_3 \subset \cdots$ of our bipartite graph G. On each graph G_k, we attempt to find a perfect matching, using an augmenting path algorithm. If no perfect matching exists, we add enough edges (of minimal additional cost) to get around a current obstruction to obtain G_{k+1}, and then repeat the process. COMAP published a detailed and accessible module [Gale, 1978] on the optimal assignment algorithm that provides an economic interpretation of the procedure. (This module considers the maximum weight assignment problem, but that formulation is equivalent to the minimum weight version via multiplication by -1.)

2.3 First-Year Courses at Macalester College

First-year seminars became mandatory at Macalester College in 1994. Every department offers one or more first-year seminars, called First-Year Courses (FYCs). The content of the seminar is discipline specific: recent seminar topics include global health, the social structure of the World Wide Web, representations of vampires, and the American West. Regardless of the topic, all FYCs develop writing skills and critical thinking. The professor also becomes the students' academic advisor, meeting with the students before the semester begins, and coordinating various social functions. Some FYCs are residential (at the request of the professor), with members of the seminar housed in close proximity of one another. Needless to say, the FYC experience is a very influential part of the student's social and academic transition to Macalester.

Each year, the responsibility for matching students to seminars belongs to the Office of Academic Programs. The assignment procedure is as follows. A typical year has roughly 500 first-year students and 35 FYCs. The incoming students read the seminar descriptions and then rank their top four choices. Academic Programs compiles these preferences, and then matches students to seminars, while adhering to some modest constraints on the distribution of students to seminars.

Prior to 2009, the assignment was performed by hand. Each student's preferences were listed on a sheet of paper, which were then organized into piles (spread on the floor!) representing each seminar. Much like the augmenting paths of the Hungarian algorithm, staff members would look for chains of swaps to improve the overall profile of the current assignment. The entire process consumed 40 person-hours before the staff was satisfied with the assignment. Figure 1 shows the historical results for these manual assignments from 2003 to 2008. On average, the Office of Academic Programs was able to place roughly 87% of the students into their first or second choice.

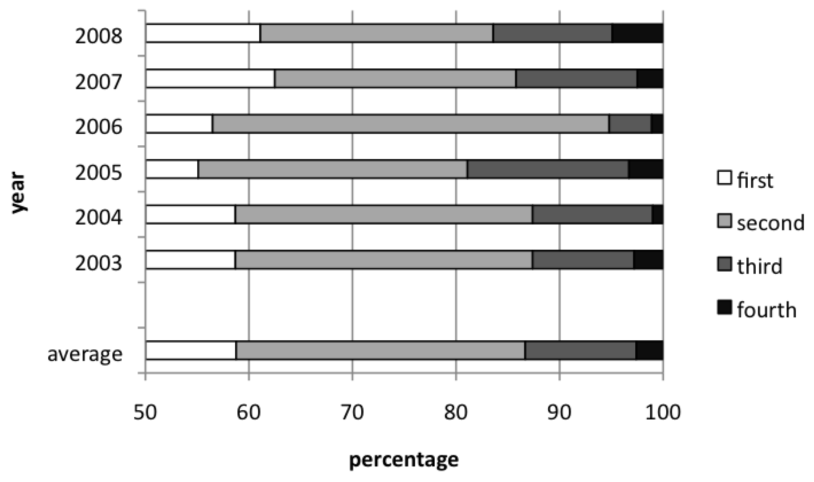

Figure 1: Distributions of manual assignments by the Office of Academic Programs, 2003–2008.

2.4 The Collaboration

We briefly recount the unfolding of the collaboration between the authors and the Office of Academic Programs. The interfacing of mathematics with a real-world application never occurs in a straight line. In particular, adopting new technology can be unsettling. Therefore, the authors purposely developed the collaboration in stages. Ultimately, this collaboration between student, faculty and staff has been a particularly rewarding experience for all involved

The project was conceived by the second author in Fall 2008 while serving as the teaching assistant for a first-year seminar in discrete mathematics. In one of the first class meetings, the professor introduced graph theory by describing the connection between graph matchings and seminar assignments. With his interest piqued, the second author suggested this topic for his project in the senior seminar on combinatorial optimization, taught by the first author. This application was a natural fit for the course, and the project was initiated. The fall semester was spent exploring and implementing matching algorithms.

In Spring 2009, the authors scheduled a meeting with the Office of Academic Programs to discuss the project. While the authors hoped that our implementation would eventually be adopted, it was crucial to "sell the client" on this solution. In the initial meeting, the authors described the project in broad strokes. Academic Programs was supportive of our effort, and agreed to evaluate the implementation once it was complete. They provided information on their constraints, their methods, and their objectives. They also agreed to proved a historical data set from 2004 (since that cohort of students had already graduated). Using these specifications and test data, the authors crafted the Mathematical Sorting Hat implementation. The Hat slightly out-performed the manual assignments from 2004, and was deemed a success.

In Summer 2010, the project became a true partnership between the authors and the Office of Academic Programs. Encouraged by the initial results, Academic Programs agreed to collaborate on the next version of the Hat. In a series of meetings, we fully mapped out specifications and requirements. Academic Programs provided feedback on the model, and further refined their constraints to accurately reflect their needs and priorities. We agreed upon the data format for input (a comma-separated value format). The second author set to work to implement these changes, creating a fully featured Mathematical Sorting Hat.

In July 2010, the Mathematical Sorting Hat was used to create an assignment for the incoming first-year students. Simultaneously, the Office of Academic Programs developed a manual assignment. Once again, the Hat's assignment was slightly better. Convinced of the benefits of the Mathematical Sorting Hat, Academic Programs opted to use the Hat's assignment. Since then, Academic Programs has fully embraced the Hat's assignments, and no longer spends any effort on manual assignments.

3 The Mathematical Sorting Hat

From the college's perspective, a successful assignment results in seminar enrollments that are a microcosm of the college, in terms of gender balance and international diversity. From the student perspective, a successful assignment will place students in the seminars that they most desire. In other words, the First-Year Course Assignment Problem (FYCAP) is a constrained optimization problem in which we try to maximize student satisfaction while respecting constraints on seminar enrollment.

The authors converted FYCAP into an assignment problem, mapping students to workers, and seminars to jobs. This presented two main challenges. First, we had to devise a bipartite graph structure that enforced all of the college constraints. Second, we had to develop an edge weighting scheme that both encoded student preferences and resulted in an assignment where most students receive their first or second choice. We refer to the resulting weighted bipartite graph as the *Mathematical Sorting Hat.*

Before describing the structure of the Hat, we list the performance results of the automated seminar assignment for the last three years. Figure 2 shows the assignments, which compare favorably to the manual assignments in Figure 1. Over the last three years, the Hat has placed

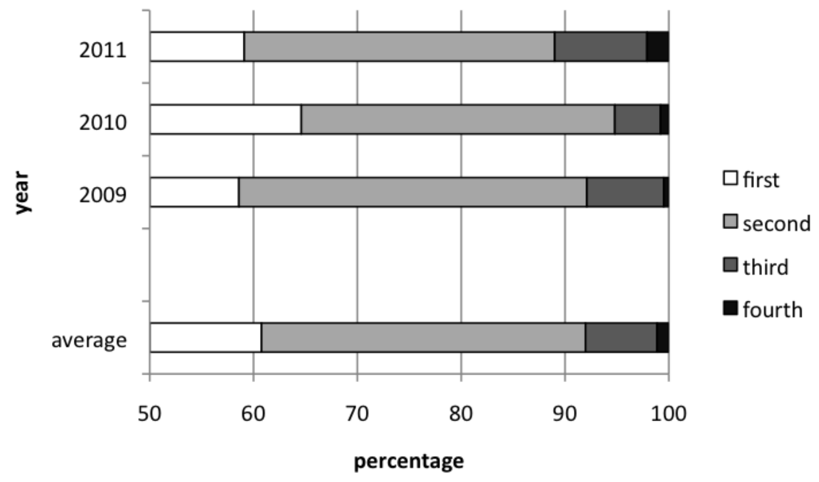

Figure 2: Distributions of assignments produced by the Mathematical Sorting Hat, 2009–2011.

roughly 91% of the students into their first or second choice. The Hat's best year (2010) is comparable to the best manual year (2006). Indeed, we expect the Mathematical Sorting Hat to produce high quality assignments. We were actually pleased to discover that the Hat's assignments were not significantly better than the historical manual assignments. This means that the hours of effort put into these assignments truly paid off.

3.1 Constructing the Mathematical Sorting Hat

We describe the bipartite graph $G = (V, E)$ with vertex partition $V = X \cup Y$, where X consists of students and Y consists of seminar seats. Suppose that we have r students and s seminars, where each seminar has enrollment limit of t students, and $r \leq st$. The r students correspond to vertices $X = \{x_1, x_2, \ldots, x_r\}$. If $r < st$, then we add $st-r$ *dummy* vertices $x'_{r+1}, x'_{r+2}, \ldots, x'_{st}$ to X. For $1 \leq i \leq s$, let $Y_i = \{y_{i,1}, y_{i,2}, \ldots y_{i,t}\}$ represent the t seats in seminar i. Let $Y = \cup_{i=1}^{t} Y_i$ be the set of all seminar seat vertices. We now have a natural mapping between FYCAP and an Assignment Problem: the students are workers and the seminar seats are jobs.

The Office of Academic Programs provided 4 constraints for the enrollment of each seminar. First, every seminar must contain at most 16 students. Second, we enforce a minimum seminar size of 10 students. Third, no seminar can have a majority international enrollment (at Macalester, international students make up 11% of the student body). The final constraint is a modicum of gender balance. Having already placed restrictions on international students, we decided to enforce gender balance only for domestic students. Therefore, in each fully enrolled seminar, 4 seats are reserved for domestic women, and 4 seats are reserved for domestic men. For seminars with fewer than 16 students, we only require 3 domestic women and 3 domestic men.

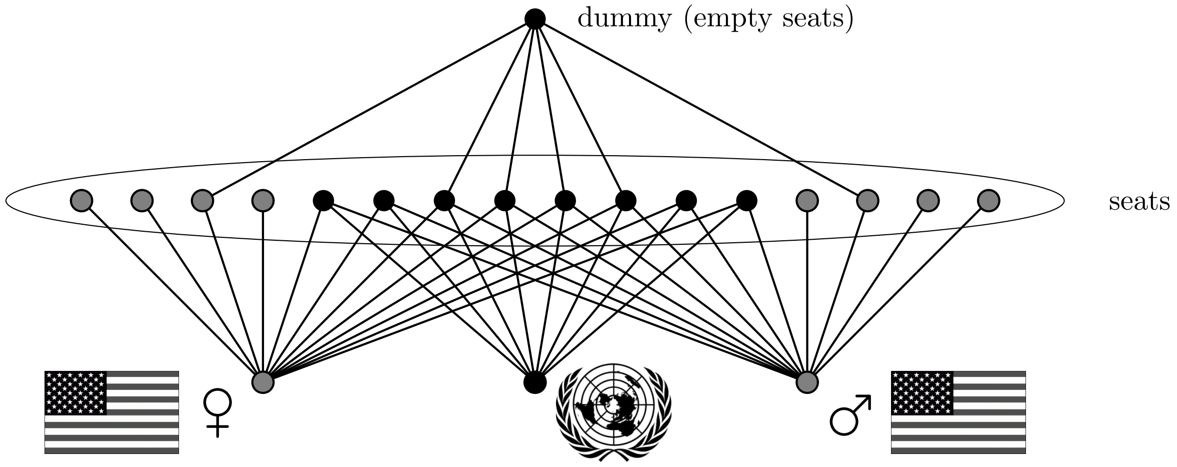

Figure 3: The edges connecting the 16 seats of a seminar to the 4 types of students: domestic female, international, domestic male and dummy students. The edge structure ensures that the seminar assignment will adhere to the college's constraints.

We enforce these four constraints via the edge structure of the graph. Figure 3 shows the local graph structure of the Mathematical Sorting Hat. It depicts the 16 seminar seat vertices for one seminar, along with one student of each type that is interested in the seminar topic. We divide the seminar seats into three categories, with 4 domestic female seats, 4 domestic male seats, and 8 generic seats. Every domestic female student interested in this seminar is adjacent to the 4 domestic female vertices and the 8 generic vertices. Likewise, interested domestic males are adjacent to 4 domestic male seats and 8 generic seats. Finally, every interested international student is adjacent to the 8 generic vertices.

Dummy students correspond to empty seminar seats. We allocate edges from dummy students to enforce the minimum seminar size of 10. Therefore, every dummy vertex is adjacent to the same 6 vertices in a given seminar. More specifically, each dummy vertex is adjacent to one female domestic seat, one male domestic seat, and four generic seats. This allocation allows the lower enrolled seminars to have a gender minimum of 3 students, while fully enrolled seminars have a gender minimum of 4 students. If no assignment exists that adheres to these size, international and gender constraints, then the matching algorithm will return a failure. Table 1 summarizes the distribution of seats to students, according to type.

	total	USA female	USA male	international	dummy	minimum enrollment
USA female seats	4	4	0	0	1	3
USA male seats	4	0	4	0	1	3
generic seats	8	8	8	8	4	4
total seats	16	12	12	8	6	10

Table 1: The number of seminar seats adjacent to each student. The existence of edges depends upon the category of the seat and the characteristics of the student. The minimum enrollment per category is total seats minus dummy seats.

We reflect on the structure of the bipartite graph $G = (X \cup Y, E)$ that we have created. By adding dummy students to fill empty seats, we have $|X| = |Y|$. The edges incident with student vertices reflect their seminar preferences. The global structure of these student edges also enforce gender and international balance. The edges incident with dummy vertices enforce minimum seminar size. The dummy edge structure also mitigates the gender constraint for lower enrolled seminars. The graph G always has a maximum matching (in terms of number of edges of the matching). This maximum matching is a perfect matching if and only if there is a valid assignment of students to seminar seats.

3.2 Modeling Student Preferences with Edge Weights

We use edge weights to capture student preferences. In this section, we describe how our choice of weighting scheme influences the global characteristics of the optimal assignment. Let (w_1, w_2, w_3, w_4) be the edge weights for the student choices. These weights satisfy $w_1 \leq w_2 \leq w_3 \leq w_4$ since the Hungarian algorithm seeks to find a minimum weight perfect matching. We give two simple examples to illustrate the impact of our weighting scheme. If we set $w_i = 1$ for $1 \leq i \leq 4$, then the algorithm will ignore student preference between acceptable seminars. If we choose $w_1 = 1$ and $w_2 = w_3 = w_4 = 100$ (or any large number), then the algorithm will maximize the number of students in their first choice, and treat the other options equivalently.

For $1 \leq i \leq 4$, let P_i denote the percentage of students assigned to their ith choice. We define the *profile* of an assignment to be (P_1, P_2, P_3, P_4). Using the average of the historical results in Figure 1 as a guide, we find that Academic Programs requires that the assignment profile obeys the following inequalities:

$$\begin{aligned} P_1 &\geq 59\%, \\ P_1 + P_2 &\geq 87\%, \\ P_1 + P_2 + P_3 &\geq 97\%. \end{aligned} \tag{1}$$

The conditions of equation 1 are the primary criterion for measuring the quality of an assignment. The secondary criterion, identified by the Office of Academic Programs, is the mitigation of the number of students in their fourth choice. We did not directly incorporate these two criteria as hard constraints to our matching problem. Instead, we considered multiple weighting schemes, both experimentally and heuristically. A weighting was considered viable if it gave an assignment that met these two criteria. These weightings were reported back to the Office of Academic Programs.

We considered two types of weighting schemes (w_1, w_2, w_3, w_4). First, we considered the naive weighting $(1, 2, 3, 4)$. Second, we considered geometric weightings of the form $(1, \alpha, \alpha^2, \alpha^3)$ for $\alpha > 1$. Geometric weightings have a natural interpretation, described below. Figure 4 presents the resulting assignment profiles for four different weighting schemes on the historical data from 2004. (The 2004 data was the initial data set provided by Academic Programs.) The results for 2004 are typical for other historical data sets. Considering these results, we see that all four weightings give assignments that satisfy the conditions of equation (1). All of the geometric weightings improve upon the manual assignment performed in 2004. The naive weighting achieves the largest percentage of first choice placements, but at the cost of assigning far more students to their fourth choice.

In order to determine the best weighting, we use our secondary criteria: the number of students receiving their fourth choice. Our geometric weightings for $\alpha = 3.5$ and $\alpha = 4.5$ reduce fourth choice placements to 1.55%, while $\alpha = 2.5$ only achieves 1.75%. Larger values of α give the same ranking as $\alpha = 4.5$, suggesting that the graph structure obstructs further improvements to the assignment. Deciding between $\alpha = 3.5$ and $\alpha = 4.5$ required consulting with the Office of Academic programs. They had to decide whether it was worth moving 1.16% down from first to second choice on order to move 0.57% up from third to second choice. The Office of Academic Programs was pleased with the trade-off for $\alpha = 3.5$, so we settled on this geometric weighting for our final implementation.

We now give an intuitive interpretation of a geometric weighting scheme. It is simplest to understand how a positive integer weight α affects the profile of an optimal matching. Therefore, we start by considering such integer α weights. After explaining that case, we then describe how sensitivity analysis led us to choose half-integer weights instead.

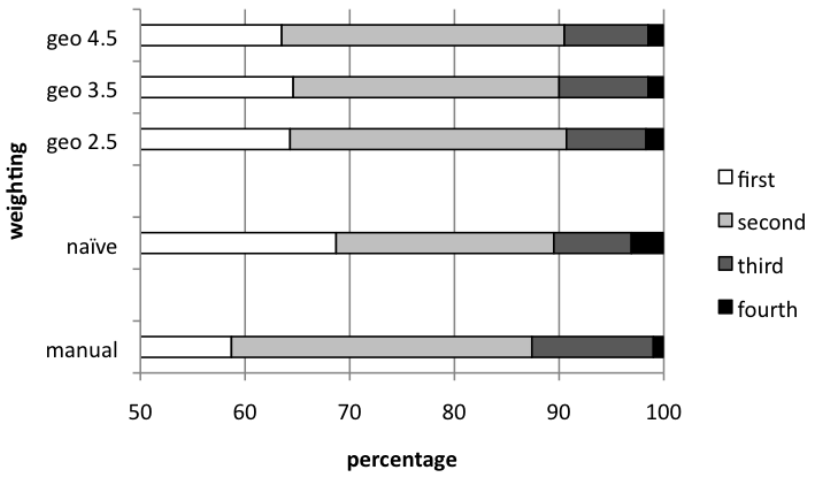

Figure 4: The profiles of assignments for four different weighting schemes for historical data from 2004. Geometric weightings are of the form $(1, \alpha, \alpha^2, \alpha^3)$ for the given value of α. The naive weighting is $(1, 2, 3, 4)$. The manual assignment was performed by the Office of Academic Programs in 2004.

Consider the weighting scheme (w_1, w_2, w_3, w_4). We may always take $w_1 = 1$ because multiplying all weights by a constant $c > 0$ does not affect the optimal matching. The geometric weighting scheme $(w_1, w_2, w_3, w_4) = (1, \alpha, \alpha^2, \alpha^3)$ has a natural interpretation in terms of trade-offs made by swapping students between seminars. Suppose we have a student in her third choice, and that we can move this student into her second choice via a series of swaps that moves k other students down from first choices to second choices; see Figure 5. The corresponding change in the weight of the matching is

$$k(w_2 - w_1) + (w_2 - w_3) = (k + 1)\alpha - k - \alpha^2 = -(\alpha - 1)(\alpha - k).$$

Therefore the swap is acceptable whenever $-(\alpha - 1)(\alpha - k) \leq 0$, or equivalently $k \leq \alpha$. This inequality captures our willingness to move $k \leq \alpha$ students down from their first to their second choices in order to move a single student up from her third choice to her second choice. The advantage to a geometric weighting is its translation invariance: we obtain an equivalent inequality for the trade-off for moving k students down from their second choices to their third choices in order to move one student up from her fourth choice to her third choice. (The naive weighting does not have this invariance. In fact, its penalty fourth choices is not severe enough.) Constraints for other swaps have similar interpretations.

Figure 5 also reveals a minor disadvantage to the simple geometric weighting $(1, \alpha, \alpha^2, \alpha^3)$ for

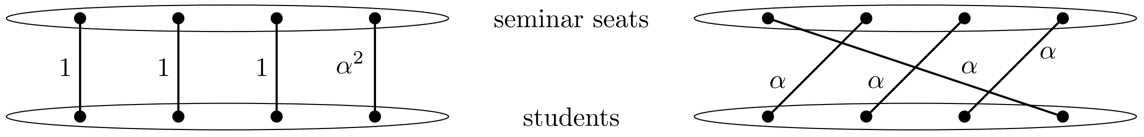

Figure 5: Two potential assignments using a simple geometric weighting scheme $(1, \alpha, \alpha^2, \alpha^3)$. The matching on the left has weight $3+\alpha^2$ and the matching on the right has weight 4α. When $\alpha < 3$, the left matching has lower weight and when $\alpha > 3$, the right matching has lower weight. For $\alpha = 3$, the matchings have the same weight.

$\alpha \in \mathbb{Z}^+$. When $\alpha = 3$, these two matchings have equal weight, so the Hungarian algorithm could return either one as its optimal assignment. To avoid such ambiguity, we guide the Hungarian algorithm to prefer the right hand matching by slightly perturbing the weights. In our final implementation, we actually used the weight $\alpha = 3.5$. This gives a consistent preference for the right matching. This aligns with our goal to reduce the number of students assigned to their third choice. A similar argument shows that we favor moving students out of their fourth choice under comparable circumstances.

Sensitivity analysis of α revealed another reason to choose non-integer weights. We performed this sensitivity analysis by running the Mathematical Sorting Hat on a range of α values on our recent data sets. The results for 2010 are presented in Figure 6. (The results for other recent years are similar.) For every $\alpha \geq 4$, the Hat returns the same matching. This suggests that we have hit a bottleneck in the underlying graph structure. Overall, the behavior with respect to α is quite stable. For $\alpha \geq 2$, the optimal matching changes at integer values for α. This suggests that the optimal matchings are, in fact, primarily shaped by the types of trade-offs exemplified in Figure 5. The results are slightly more unstable for smaller α, with an additional change at $\alpha = 1.5$, probably due to beneficial 3-to-2 swaps. The stability of our model may be related to the relative small size of the data set. For larger data sets, the stability for $\alpha \geq 2$ might decrease due to additional opportunities for more exotic beneficial swaps. In conclusion, our sensitivity analysis suggests that four our data sets, choosing half-integer values of α gives the most stable results when $\alpha \geq 2$, while integer α value results give the most unstable results. Therefore, we use half-integer values for α.

The last edge type to consider are those incident with dummy vertices. An assignment of a dummy vertex corresponds to an unfilled seat in the seminar. The dummy vertices have no preferences, so all of their edges will be assigned the same weight w_5. We always prefer to assign a real student to a seat, so the dummy weight should be much larger than w_4 to ensure

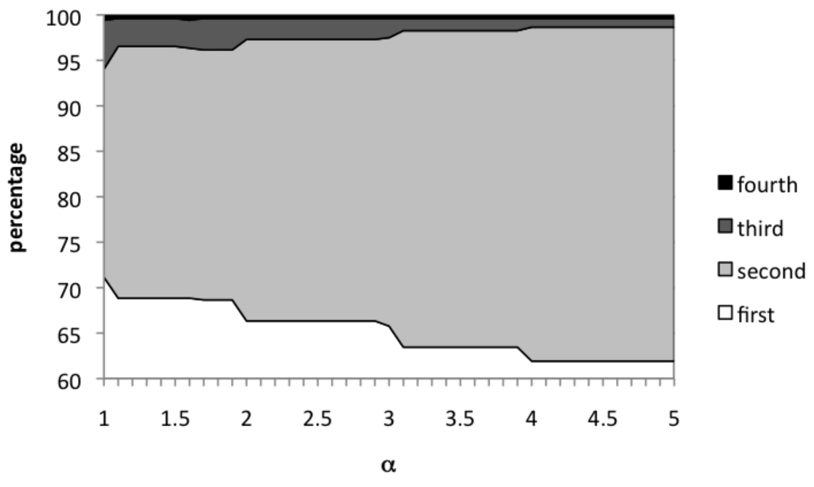

Figure 6: Sensitivity analysis for weight α on the 2010 data set.

that a real student is always given priority. For a near-geometric weighting, setting $w_5 = \alpha^4$ is sufficient.

In working with Academic Programs, we focussed on $\alpha = 2.5$ and $\alpha = 3.5$. These values do result in high quality matchings. Moreover, this choice was instrumental in making Academic Programs comfortable with trusting the Hat. Indeed, these values were chosen because they matched the historical assignment methodology. Client satisfaction is the ultimate requirement, and we were happy to provide a solution that they could understand. Now that Academic Programs has embraced the Hat, we may provide a broader spectrum of weights for them to choose from in the future.

3.3 Running the Sorting Hat

We now describe how we use the Mathematical Sorting Hat code in practice, and discuss how we handle the produced assignment. The Matlab/Octave code, along with a more detailed user's manual, is available on the COMAP website.

The Sorting Hat requires two inputs: the location of the data file, and the weight α. The code constructs the weighted bipartite graph model described in this section, with one modification. We actually create a complete bipartite graph. Edges from a student to an undesired seat are given weight α^5. The advantage to this modification is that the Hungarian algorithm always returns an assignment. Indeed, since we now have a complete bipartite graph, there are $n!$

perfect matchings. The Hungarian algorithm will return one of minimum weight.

After the algorithm completes (in time $O(n^3)$), the Sorting Hat organizes the the results into two human-readable formats: one list organized by student name, another list organized by class name. Next, we decide (manually) whether is assignment is acceptable. We say that an assignment is *student-valid* when every student is assigned to one of her preferred courses. In order to assess this validity, the Sorting Hat produces a third output: a 5-entry vector of the form (first, second, third, fourth, other). The first entry contains the number of students assigned to their first choice. The second, third and fourth entries are populated similarly. The final entry contains the number of students who were not assigned to any of their specified choices. This final entry is zero if and only if the assignment is student-valid.

We say that the assignment is *college-valid* if it adheres to all of the college's constraints (class size, gender balance, international balance). It is possible for an assignment to be student-valid and not college-valid (or vice versa). For example, suppose that there is a particularly "unpopular" course, and only eight students include that course among their choices. In this case, those eight students will be assigned to that course, but we will still be below the minimum enrollment of ten students. When this occurs, there is no way to produce an assignment that is both student-valid and college-valid: we must either accept the smaller seminar enrollment, or force students into a course they do not want. As you would expect, Macalester would choose the former option.

If there is an assignment that is both student-valid and college-valid, then the Sorting Hat will find it. Indeed, when α is large enough, the penalties for failing to meet both criteria drive the Hungarian algorithm to find this assignment. In the past three years, the Sorting Hat has always returned a student-valid assignment. However, one year it returned an assignment that was not college-valid. In particular, one seminar did not satisfy the gender balance requirement because not enough men were interested in the course. In this case, Academic Programs allowed this exception to their gender balance constraint (since placing students in unwanted courses is considered unacceptable). In the future, Macalester will continue to bend its constraints in situations like this. However, if we ever fail to find a student-valid assignment, Academic Programs will resolve this by increasing the enrollment cap in a seminar to create a seat for the unmatched student.

In conclusion, we reflect on the necessity of fourth choice preferences. Indeed, it is natural to wonder whether an assignment exists so that every student gets one of their top three choices. If such an assignment exists that is also college-valid, then the Sorting Hat will find it for large enough α. Historically, this has not yet happened. It is easy to assess the impact of the college constraints on these fourth place assignments: we simply run the Sorting Hat on the choice data, altered so that all fourth choice preferences removed. If the resulting assignment is student-valid, it will be of lower weight than the assignment from using all four choices. However, this lower weight assignment will certainly be even further from conforming with the college's constraints. So it is "better" for students and "worse" for the college. Regardless, it is interesting to perform an assignment using only the first three choices. The result gives us a way to measure the impact of the college's constraints on student satisfaction.

4 Conclusions

We have described the conversion of the First-Year Course Assignment Problem at Macalester College into a classical Assignment Problem. We developed a weighted graph capturing student preferences and enforce college constraints. Through experimentation on historical data, we developed an edge weighting scheme that produces assignments profiles that place 91% of the students in their first or second choice. The Office of Academic Programs has been thrilled with these results, both in terms of man hours saved, and in the quality of the assignment.

Looking to the future, we hope to improve our assignment process and produce higher quality results. Along with adopting the Mathematical Sorting Hat, the Office of Academic Programs now collects student preferences electronically. This combination opens the door to handling more robust information about student preferences. For example, rather than a simple numerical ranking, we could use a sliding scale to allow students to express interest in a seminar. This would allow us, for example, to distinguish between strong a weak preferences between a student's first and second choice. We hope to work with Academic Programs to investigate such a strategy in the coming years.

More broadly, other schools might consider adopting an automated assignment procedure like the Mathematical Sorting Hat. For Macalester, we devised a bipartite graph model encoding the constraints required by the college. Essentially, the ratio of seminar seats to constraints was favorable enough that we could encode these constraints via the edge structure of the graph. Other schools would have to develop a graph structure that reflects their specific priorities. It is possible that their constraints are too complex to accurate model in a bipartite graph. In this case, they should consider using a binary integer program to perform the assignment (cf. [Bertsimas and Tsitsiklis, 1997]). Such programs offer great flexibility. While their solution methods are more complicated, there are many integer program solvers available. Schools must also be careful when developing their model, since too many constraints may lead to an infeasible program with no possible solutions. Therefore, additional constraints must be added incrementally and judiciously.

Acknowledgments. We are grateful to Tom Halverson for introducing this application to the second author. We thank Ann Minnick and Brenda Piatz, from the Macalester Office of Academic Programs, for their openness to embracing this project, and for sharing their historical data with us. Finally, we thank the anonymous referee for suggested improvements to both the exposition and the accompanying code.

5 About the Authors

Andrew Beveridge earned his BA in mathematics from Williams College (1991) and his PhD in mathematics from Yale University (1997). After spending time in both academia and industry, he is currently an Assistant Professor at Macalester College. His main research area is probabilistic combinatorics, where he is particularly interested in the interplay between randomness and strategy. His hobbies include volleyball, running and playing bass guitar.

Sean Cooke graduated from Macalester College in 2009, with majors in Mathematics and Computer Science and a minor in Hispanic Studies. He is interested in applied mathematics, computation and combinatorial optimization He is currently a chef at Sodexo, and plans to pursue a quantitative MBA. He is also a freelance graphic designer and a musician who plays a wide array of musical styles from across the globe. He plays guitar, drum set, Latin and Brazilian percussion, and he is learning saxophone. He released his own album in December 2011.

References

[Abdulkadiroğlu et al., 2005] Abdulkadiroğlu, A., Pathak, P. A., and Roth, A. E. (May 2005). The New York City high school match. *American Economic Review, Papers and Proceedings*, 95(2):364–367.

[Abdulkadiroğlu and Sönmez, 1999] Abdulkadiroğlu, A. and Sönmez, T. (October 1999). House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260.

[Abraham et al., 2007] Abraham, D., Blum, A., and Sandholm, T. (2007). Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *ACM Conference on Electronic Commerce*.

[Bertsimas and Tsitsiklis, 1997] Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*. Athena Scientific.

[Cook et al., 1998] Cook, W., Cunningham, W., Pulleyblank, W., and Schrijver, A. (1998). *Combinatorial Optimization*. John Wiley & Sons.

[Emspak, 2010] Emspak, J. (2010). First kidney transplants using new matching algorithm. `http://www.ibtimes.com/articles/92162/20101214/first-kidney-transplants-using-new-matching-algorithm.htm`.

[Evans and Minieka, 1992] Evans, J. and Minieka, E. (1992). *Optimization Algorithms for Networks and Graphs*. CRC Press, 2nd edition.

[Gale, 1978] Gale, D. (1978). The optimal assignment problem. *COMAP Module 317*.

[Gale and Shapley, 1962] Gale, D. and Shapley, L. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69:9–15.

[Khuller et al., 1991] Khuller, S., Mitchell, S. G., and Vazirani, V. V. (1991). On-line algorithms for weighted bipartite matching and stable marriages. *Automata, Languages and Programming*, 510:728–738.

[National Resident Matching Program, 2011] National Resident Matching Program (2011). National resident matching program. `http://www.nrmp.org/res_match/index.html`.

[Roth and Peranson, 1999] Roth, A. E. and Peranson, E. (1999). The redesign of the matching market for american physicians: Some engineering aspects of economic design. *American Economic Review*, 89:748–780.

[Rowling, 1998] Rowling, J. K. (1998). *Harry Potter and the Sorcerer's Stone*. Scholastic.

[StateUniversity.com, 2011] StateUniversity.com (2011). College seminars for first-year students – types of first-year seminars, course objectives and content, pedagogy and staffing, instructor development.

`http://education.stateuniversity.com/pages/1862/College-Seminars-First-Year-Students.html`.

[West, 2000] West, D. B. (2000). *Introduction to Graph Theory.* Prentice Hall, 2nd edition.

A Implementation Notes

In this appendix, we address some of the practical issues in using the Mathematical Sorting Hat to assign students to seminars. Topics discussed include fairness, repeatability and managing incomplete data sets.

A.1 Dependence on student ordering

The Mathematical Sorting Hat returns a minimum weight assignment. However, there is no guarantee that this assignment is the unique minimizer. (In an extreme case, if 64 students gave identical rankings to 4 available courses, then every matching would be minimal.) We believe that most real world data will have multiple optimal assignments. Therefore, we must address factors that influence the optimal assignment returned by the Hat.

The most important observation is that the Hungarian algorithm gives an advantage to the students that are processed first. Indeed, in each iteration, the assignment algorithm makes partial assignments and then tries to improve upon them via alternating paths. A student processed early on will be placed into her first choice seminar. She will be moved from this seminar only if there is an augmenting path that leads to a matching of strictly lower weight. On the other hand, if a student desired a seat in an already full seminar, then she will only be assigned there is there is augmenting path that improves from the current assignment.

In order to mitigate this effect, we preprocess the student data by randomizing the order. We perform this randomization once, so that we may perform repeated runs on the data.

A.2 Performing multiple runs

We always perform multiple runs on the data. Early runs are used to validate the input data. We typically perform multiple runs on the final data. In consultation with Academic Programs, we tweak the constraints and then observe the impact on the quality of the matching. This allows the Office of Academic Programs to respond to specific characteristics of the data set.

For example, if we achieve a particular good assignment (say 95% in their first or second choice), then Academic Programs may be interested in seeing the impact of greater gender balance constraints. If the trade-off is not too bad, they may decide that this new assignment is preferable. On the other extreme, they may want to understand the negative effect of a particular constraint. In that case, we remove the constraint to get a base-line comparison.

A.3 Data validation

Prior to running the Mathematical Sorting Hat, the data must be validated. We perform validation as a separate step, so that we may tolerate unusual students, as described below. During validation, we check that every student has both gender and international status, and has ranked four courses. Students who rank fewer than four courses are at an advantage. Indeed, a student can only be assigned to one of the courses that she ranked. All data errors are reported to Academic Programs, who then gives instructions for resolution.

A.4 Unusual students

In reality, there may be students who must be treated exceptionally. It is the prerogative of Academic Programs to determine how to handle these students. For example, a transgendered student does not fit into one of our designed categories. As another example, there may be a student who must be assigned to a particular seminar, or who does not have the pre-requisites for some of her choices. If a student must be assigned to a particular seminar, then it is easy to do so: only provide that one choice for the student.

Our code requires that all entries are populated. So a transgendered student must be assigned as 1 (female) or 0 (male, or perhaps 'not female'). When a student has ranked fewer than 4 courses, the remaining ones are marked as 'None.' Students with missing preferences must be okayed by Academic Programs.

A.5 Missing student preferences

Every year, a handful of students do not submit preferences before the deadline. These students are removed from the data set, and the assignment is performed. When the Office of Academic Programs is able to get in touch with those students, they are allowed to choose from the seminars that still have unfilled seats.

A.6 Is the Hat strategy-proof?

Given a data set of student preferences, the Mathematical Sorting Hat finds an optimal matching. However, we know very little about how the students go about ranking their courses. From the student perspective, first-year seminar assignment is a game in which they are competing against all other students. Is this game *strategy-proof*, meaning that the student has no incentive to lie about their true preferences? Certainly a student who ranks a very popular course as her fourth choice has virtually no chance of being assigned to that course. Could a student try to take advantage of this behavior to increase her chance of being assigned to her first choice? Or would this strategy have a greater chance of leaving her unassigned entirely, and forced to choose from the remaining (undesirable) slots? This is an interesting open question, whose answer might depend on the weighting scheme.