Meetings, Bloody Meetings!

Joshua M. Horstman
Jamie Kawabata
James C. Moore, IV
Rose-Hulman Institute of Technology
Terre Haute, IN 47803

Advisor: Aaron D. Klebanoff

Introduction

We present a model and three algorithms for solving the problem of scheduling An Tostal Corporation's upcoming board meeting. To determine how well a schedule will fit An Tostal Corporation's needs, we establish a badness function based on assumptions as to what kinds of schedules are desirable. These assumptions include that a good mix of people in meetings is pivotal for effective idea-sharing.

The algorithms that we compare are based on random selection, greedy assignment, and greedy assignment followed by hill-climbing. The random algorithm places board members at random. The greedy algorithm assigns one member at a time by examining which possibility is locally optimal, hoping for a solution that is globally optimal. The greedy algorithm with hill-climbing enhances the greedy algorithm's effectiveness by tweaking the schedule in a manner that will cause the schedule to edge closer and closer to optimality.

We also provide a simple algorithm that will allow a secretary to handle any unforeseen additions or cancellations. This algorithm is designed to alter as few board members' schedules as possible while still obtaining a good mix of people.

Finally, we summarize the strengths and weaknesses of the algorithms presented, present a sample solution for use by the An Tostal Corporation, and conclude that the "greedy-twiddle" algorithm is most effective and provides a very good solution nearly every time.

The UMAP Journal 18 (3) (1997) 321–329. ©Copyright 1997 by COMAP, Inc. All rights reserved. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

Assumptions and Justifications

- All discussion groups in a given session should be of approximately the same size. A discussion group that is too large will not facilitate good discussion. If any of the discussion groups are too small, then some of the discussion groups will be too large (since there are a fixed number of discussion groups and board members), and we will have the same problem.
- Since concurrent discussion groups will be of roughly equal size and we are to put a proportionate number of in-house board members in each discussion group, each concurrent discussion group should have approximately the same number of in-house members.
- All board members are treated equally, with the exception of in-house employees as outlined in the problem statement. There are no special needs considered in the assignment of schedules, such as a board member desiring to avoid a particular individual.
- All of the senior officers are present for the three morning meetings.
- The schedules will not be adjusted for board members who are late or have to leave during the middle of the day.
- When the secretary is performing a last-minute change to the schedule, it is desirable to minimize the number of members who experience a schedule adjustment. That is, a change that completely rearranges one member's schedule is better than a change that slightly alters many members' schedules.

Defining the Model

A natural model is a hypergraph in which each vertex corresponds to a board member and each edge corresponds to a discussion group. However, the added constraints, such as the in-house board members' even distribution and the limitation that each board member see each officer at most once, disrupt the structure of the hypergraph enough to render many of the well-known theorems inapplicable. On the other hand, some of the matrices derived from the hypergraph model, such as the adjacency matrix and the incidence matrix, are quite useful for our computations.

With this as the underlying model, each board member corresponds to a row and a column of the adjacency matrix, and to a row of the incidence matrix. A column of the incidence matrix corresponds to a discussion group. Each board member is represented by a number from 1 to 29, with 1 through 9 representing the in-house board members (corporate employees).

A good schedule is one that does a good job of meeting the following criteria:

- The total number of board members in any two concurrent discussion groups does not differ by much.
- The number of in-house board members in any two concurrent discussion groups does not differ by much.
- In the morning sessions, no board member is in multiple discussion groups led by the same senior officer.

We design our algorithm so that every schedule produced satisfies the third condition; so we attempt to devise a schedule that does a good job of meeting the first two criteria. We measure how good a schedule is by defining a function that assigns a "badness" to each schedule. First, we define an *encounter* as one instance of two board members being in the same discussion group. We want our badness function to penalize, by increasing the badness, a schedule that contains a pair of board members who have too many encounters. Making sure that no board members have too many encounters automatically ensures that no board members have too few encounters. Since there is a fixed number of encounters (as long as the discussion group sizes stay constant), then if one pair of board members has too few encounters, it is only because another pair has too many. We define e_i to be the number of pairs of board members who encountered each other i times. A reasonable function for the badness b of a particular schedule S is

$$b(S) = \sum_{i=2}^{\infty} e_i \cdot 4^{i-2}.$$

By making the badness penalty increasingly higher for each additional encounter between a pair of board members, we create a function that is minimized when every pair of board members has the same number of encounters. This badness function is not good enough, though, since such a function also must take into account the distribution of the in-house board members. We will say that all schedules for which the numbers of in-house members in any two concurrent discussion groups differ by no more than one are equally good. However, a very heavy penalty will be imposed for each discussion group that contains a number of in-house board members that is more than one away from the average number of in-house board members in a discussion group. Define d to be the number of discussion groups in the entire schedule that fall into this category; then we can revise our badness function to

$$b(S) = 1000d + \sum_{i=2}^{\infty} e_i \cdot 4^{i-2}.$$

This penalty prevents deviation from an even distribution of in-house board members. Minimizing the badness of a schedule will take care of the first two criteria of an optimal schedule referred to above. The minimum theoretical value for $b(S)$ is 129, calculated using the Pigeonhole Principle.

Random Selection Algorithm

There is a huge search space, containing more than a googol (10^{100}) of schedules, so a brute-force attack is out of the question. To provide a basis for comparison, we see how effective a random selection algorithm is. This algorithm assigns each board member to a random discussion group in each session, making sure that no board member is in multiple discussion groups led by the same senior officer. Although we impose no limit on the size of a discussion group, we hope that a random algorithm will distribute the board members evenly. We implemented this algorithm on a computer and ran it 1,024 times. **Figure 1** displays the badness of these executions. The results are not good.

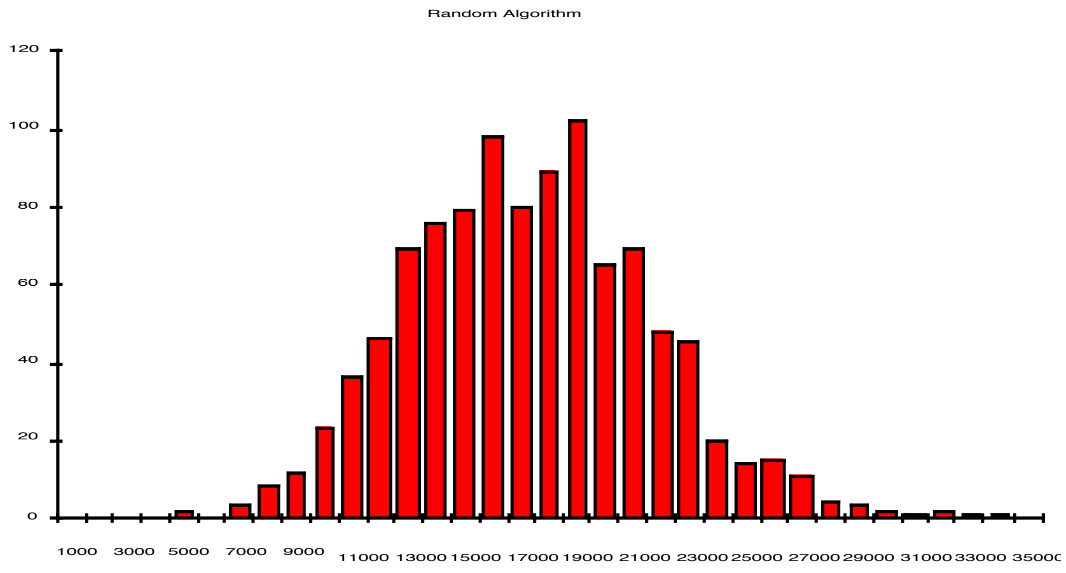

Figure 1. Badness results for random assignments.

Greedy Algorithm

We sought a heuristic requiring a minimum amount of backtracking, and a greedy algorithm seemed logical. We assign the in-house board members separately before assigning the rest of the board members. Each board member is placed in the first available position that minimizes the number of encounters that board member has had with any other board member. The algorithm, shown in **Figure 2**, works by assigning all of the in-house board members to each session, then going back and filling each session with the remaining board members. The order in which board members are placed and the order in which the various discussion groups are tried is random. This feature facilitates distributing the board members evenly, by eliminating regular patterns that may occur when placing the board members in the same way every time.

```
for member from 1 to 9 (in random order)
   for session from 1 to 3
     select the groups in which member has never been
     of them, choose the groups containing the fewest other members
     of those, choose the group containing members with whom member has been
       the fewest times
     place member in this group
   for session from 4 to 7
     select the groups containing the fewest other members
     of those, choose the group containing members with whom member has been
       the fewest times
     place member in this group
for session from 1 to 7
   for member from 10 to 29 (in random order)
     set greedlevel to 0
     select a group not led by an officer already encountered by member
     repeat
       does this group have a member with more than greedlevel encounters?
         If no, place member in this group
         If yes, select a group that hasn't been tried yet
           If every group has been tried, increment greedlevel and consider all
            groups untried
     until member is placed
```

Figure 2. Greedy algorithm.

Figure 3 shows the badness from 1,024 runs of the greedy algorithm. The greedy algorithm is a *dramatic* improvement over random assignment—all badnesses are below 300, while the best that random assignment could do was more than 5,000. However, there are many final schedules that the greedy algorithm can never reach. For example, in first several sessions, no board member will see any other more than once, since the greedy algorithm won't place two board members who have already encountered each other together in a meeting until it is unavoidable. The best schedule, on the other hand, may have a pair of board members who encounter each other in both the first and second sessions; but the greedy algorithm will never find it.

Greedy-Twiddle Algorithm

We can do better by using hill-climbing together with the greedy algorithm. This means that we make small changes to our schedule, determine whether the slightly modified schedule is better or worse, using our badness function, and proceed making small adjustments until we can do no better. The greedy algorithm helps a lot, in that we start our hill-climbing from a schedule that is much better than a random assignment. We call the small changes *twiddles*. Each twiddle consists of swapping two board members between meetings in a

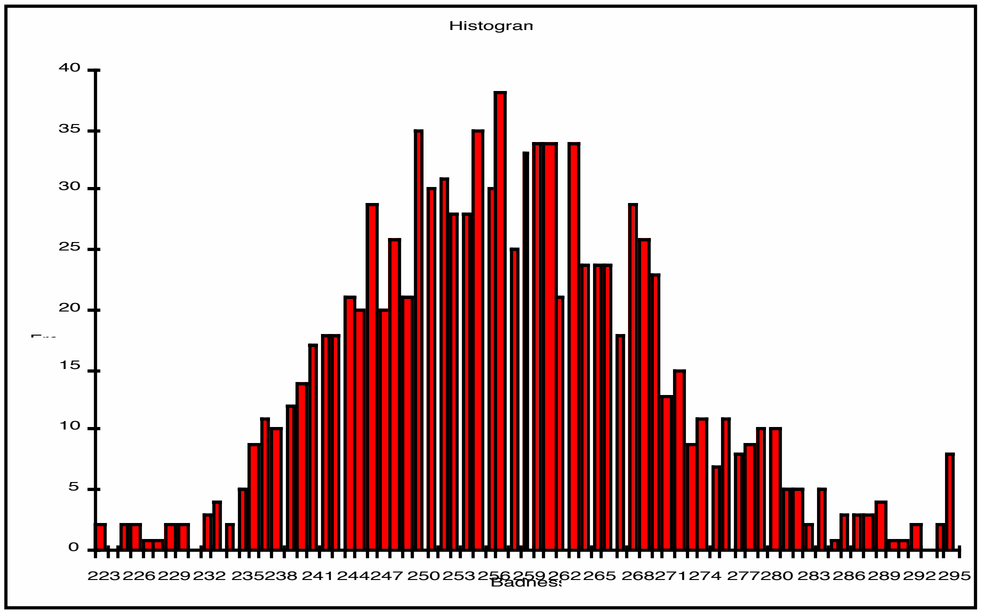

Figure 3. Badness results for assignments made by the greedy algorithm.

session or simply moving a board member from one meeting to another. After twiddling, we compute the badness function and determine if the twiddle made our schedule better or worse. If it made it worse, we undo it. We continue until no swap or move improves our schedule.

Figure 4 shows the badness of 679 runs of the greedy-twiddle algorithm, which shows significant improvement on the simple greedy algorithm—all badnesses are below the best that the greedy algorithm could do.

Table 1 shows our final recommendation to the An Tostal Corporation on scheduling their meeting; it is the best schedule from the 679 runs mentioned above.

Secretary's Algorithm

To maintain the near-optimality of the schedule while changing as few members' schedules as possible, the secretary will do the following when performing last-minute changes:

- Determine the net change in the number of in-house members who will be attending and the net change in the number of out-of-house members who will be attending. If you can pair up an in-house member who canceled and an in-house member who added, then just give the added member the schedule originally assigned to the canceled member, and similarly for out-of-house members. Do this for as many pairs as possible.
- Treat the situation for in-house members as follows:

Table 1.
Recommendation on scheduling.

Session	Members
Morning 1	1 3 10 12 26
	2 8 17 21 25
	6 13 22 28
	5 11 18 23 27
	4 14 15 19 24
	7 9 16 20 29
Morning 2	7 11 13 17 24
	1 4 15 22 27
	8 9 16 18 26
	6 10 14 20 25
	3 5 21 28 29
	2 12 19 23
Morning 3	4 5 25 29
	6 7 11 19 26
	3 15 17 20 23
	9 12 13 21 24
	2 10 16 18 22
	1 8 14 27 28
Afternoon 1	2 3 6 18 24 26 27 29
	8 9 11 12 14 15 22 25
	1 5 13 16 19 20 21
	4 7 10 17 23 28
Afternoon 2	5 6 7 12 15 18 21
	2 4 11 14 20 26 28
	1 9 17 19 22 24 29
	3 8 10 13 16 23 25 27
Afternoon 3	5 8 20 22 23 24 26
	3 7 9 18 19 25 28
	1 2 10 11 13 15 29
	4 6 12 14 16 17 21 27
Afternoon 4	5 9 10 15 17 26 27
	4 8 12 13 18 19 20 29
	1 6 11 16 23 24 25 28
	2 3 7 14 21 22

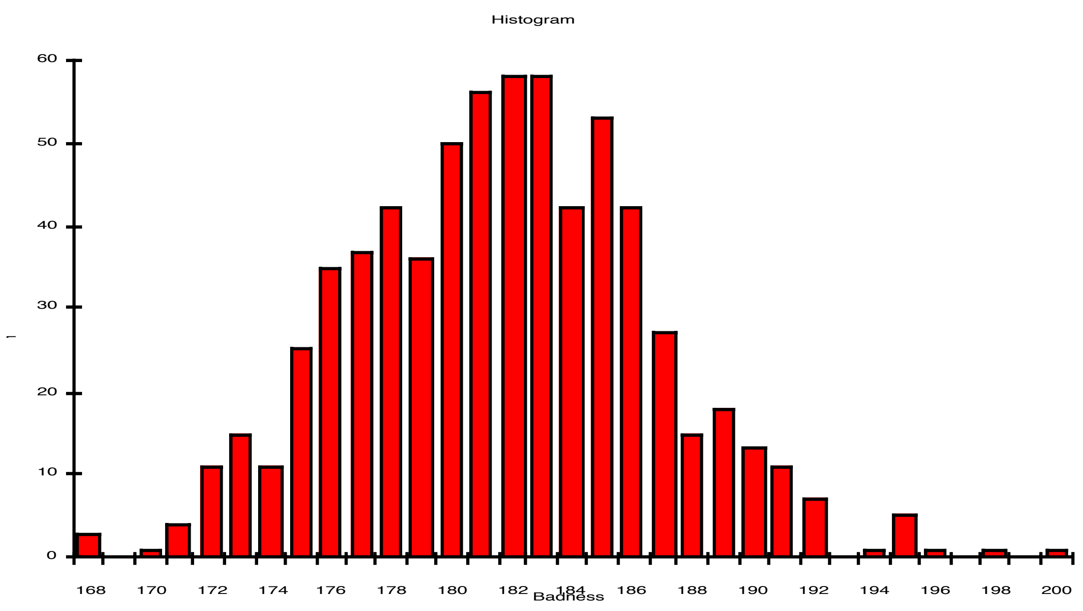

Figure 4. Badness results for assignments made by the greedy-twiddle algorithm.

- If there are still in-house cancellations that could not be paired with in-house additions:
  - Remove from each discussion group the member(s) who canceled.
  - If this makes the number of in-house members in a discussion group disproportionately low, move an in-house member from a group with more than the optimal number of in-house members to this group. Move as few members as possible, choosing first from members whose schedules have already been changed. Repeat for each discussion group.
- If there are still in-house additions that could not be paired with in-house member cancellations: For each session, place each added member in a discussion group by following these steps:
  - Eliminate all discussion groups led by a senior officer whom the member has already encountered.
  - From the discussion groups remaining, eliminate any group that has more in-house members attending than any other remaining discussion group.
  - For each remaining group, determine which member has had the most encounters with the member to be added. Choose the group for which this number of encounters is the lowest.

- Treat the situation for out-of-house members in analogous fashion.

Strengths and Weaknesses

We detail the the strengths and weaknesses of the various algorithms in **Table 2**.

Table 2.
Strengths and weaknesses of the algorithms.

Algorithm	Strengths	Weaknesses
Random	very simple, fast, requires no computer	bad schedules uneven group sizes
Greedy	relatively simple, fast on a computer	mediocre schedules, need computer for large problems
Greedy-twiddle	very good schedules	slow, difficult to program
Secretary's	secretary could do it by hand, many schedules stay unchanged	not as good as completely recomputing the schedule

Other Applications

Although the problem that motivated this model has many details unique to its situation, the model that we have developed has many other applications, such as a similar meeting with different numbers of board members or discussion groups, and also for other situations.

Consider, for example, the case of Mardi Gras Junior High School. The 120 eighth-grade students there are required to take four core classes: English, mathematics, history, and science. All four classes are offered during each of four class hours. The students also take one of six concurrent home rooms and one of six concurrent study halls. They must attend each of the core classes exactly once, but a student may have the same teacher for home room and study hall.

The school board believes that the students' educational experience can be enhanced by ensuring that the students are mixed as well as possible. That is, every student should have roughly the same number of classes with every other student. Furthermore, the board feels that the 30 honors students should be distributed so that each class (including home room and study hall) has a proportionate number. The model and algorithms developed for use at the An Tostal Corporation could be applied with minimal changes at Mardi Gras Junior High School.