Use of seeding programs

This document describes the seeding software written by Hans van Staveren. Let me start by immediately saying that these programs are tools, meant to be integrated into a wider range software, or used by people not afraid of CSV files and command windows or even Linux. There is no user interface whatsoever. That should get the disclaimer out of the way…

The target of this document is a tournament director with a reasonable amount of skill in the use of computers. If you get nervous by a prompt in a command window, please ask for outside help.

Seeding basics

The process of seeding for a tournament is divided into two steps.

1. Making groups of contestants, of similar average strength, and balanced in other criteria, such as country for example.
2. Numbering the contestants in these groups to avoid late meetings of same criteria contestants, such as country.

These two steps are implemented by the two main programs, called *seeding* and *numberlines*.

Both programs read from standard input a file, sometimes containing some numbers at the start, followed by a list of contestants and write on standard output a list of contestants in a different order. If you don't know what I mean by standard input or output, here is a short example. Suppose the input of the program seeding is contained in a file called input.txt, and you want the output written to a file called output.txt, then in a command window (or of course Unix, Linux or whatever) you type: "*seeding <input.txt >output.txt*" and the program will do it.

The information about contestants is always in the same format: a comma separated list of four values:

1. A string, representing pair-id1. This could be their names, or a number identifying them in your database, or whatever. The program will not interpret this value.

2. A string, representing pair-id2. The same goes, no interpretation. The reason there are two values is to enable you to use one value as a database-id, and the other as the real name(s). The seeding program has a -w flag (Wheelchair mode) that interprets a number at the start of this string as the group the contestant should be placed.

3. A number greater than 0, indicating strength. The program does not interpret this, although it is assumed that the lower the number, the stronger the contestant. Otherwise, the number has no meaning. If you use strengths 1,2 and 3, or you use 200, 600 and 888 the function is the same.

4. A string, representing a criterion, such as country or club. There is no interpretation of the string, but two contestants having the same string here will be put into one group.

So as a simple two-line example:

2345,Bocchi – Duboin,1,Italy
7654,van Staveren – Beks,999,Netherlands

The output of the programs is in lines of the same format.

The *seeding* program

The purpose of the program is to take all contestants for a tournament and divide them into groups based on two criteria: strength, and other. The groups need not be the same size. The program will take care to make all groups of roughly the same average strength and will within this limitation try to balance the other criteria over the groups. So, if you use country as the other criterion, and the file contains about 30% Polish pairs, the program will try to make each group contain roughly 30% Polish pairs. To make this possible the program needs some leeway in the strength. There should be several contestants of the same strength. If not, for example you have 100 contestants with strengths from 1 to 100, the program cannot balance. It is suggested to have no more than about 5 different strengths. If you use *seeding* to seed a semi-final of 60 pairs into four groups of 15, based on qualifying stages, you could give the first 4 pairs strength 1, the next 8 pairs 2, the next 16 pairs 3, the next 32 strength 4. This will give a good seeding and gives the program room to balance the countries.

Now for the input format of *seeding*. The program needs a list of numbers:

1. The number of groups you want.
2. For each group the size

So, for the example of the semi-final above the input file will start with 5 numbers:

4
15
15
15
15

And this will be followed by a list of 60 contestants.

As I have said, the groups need not be the same size. Suppose you have 56 contestants, and wish to seed for 5 groups of 11, and a pivot for the Howell. Then the input would be 6 1 11 11 11 11 11, but each number on a separate line. The effect of a group of 1 by the way is that it will get an average strength contestant.

Seeding will also accept command line arguments for the group sizes, so the above example can be replaced by giving the arguments 1 and 5x11 on the command line. This is often easier, but you must choose one of the two methods.

After *seeding* is run it will make files seededXX.txt for XX from 01 to number of groups, with each file containing the contestants for that group. When given the -t flag all groups will be written on standard output, one after the other(this is for backward compatibility with a previous version).

Within the groups by default the contestants are put in a random order. The -s flag asks for the contestants to be ordered by strength. In general, random order is better.

Further ordering within the group is left to the organizer. You could use *numberlines*, described below, or you could leave the random order.

The *numberlines* program

The second phase of seeding is numbering the contestants within the group. You can do that by randomizing for a simple tournament, but you can also let *numberlines* do it for you. The function of *numberlines* is to take a description of a movement, and a bunch of contestants, and number the contestants in such a way that contestants having the same criterion will meet each other at the earliest opportunity. If you run a one session tournament it is easy, if it is multi session you must decide. If you want the pair numbers in the group the same for all sessions, you must run *numberlines* for the final session and backtrack from there. You can also run *numberlines* for each session, to get early meetings in all sessions, but that gives pairs different numbers per session, and contestants (and sometimes even organizers) tend not to be able to cope with that. The decision is yours.

It must be stated here that *numberlines* knows nothing about movements. The good news is it can deal with any movement, the bad news is that you must specify in awful detail what the movement looks like. Simple movements, like a Mitchell give *numberlines* an easy task. It will be finished in a split second and give perfect results. But endless barometers for example are more difficult. The program will take more time (it is currently set to take a maximum of 15 minutes!) and the results, although reasonably good, will still have countrymen meeting later than you like. This is unavoidable, these movements are intrinsically difficult in this respect.

Ok, what does the input to *numberlines* look like? Again, it is a series of numbers followed by the lines of the contestants. The numbers:

1. Number of rounds in the movement
2. Number of meetings per round
3. Number of groups of contestants in the meeting
4. For each such group, the size
5. For each round, for each meeting in this round, the numbers of the two contestants meeting

Let me give a simple example, say a Howell 8:

```
7
4
1
8
1 8 2 7 3 6 4 5
2 8 3 1 4 7 5 6
3 8 4 2 5 1 6 7
4 8 5 3 6 2 7 1
5 8 6 4 7 3 1 2
6 8 7 5 1 4 2 3
7 8 1 6 2 5 3 4
```

The number of rounds is 7, there are 4 meetings per round, there is one group of size 8. In the first round there is a meeting between pairs 1 and 8, 2 and 7, 3 and 6 and finally 4 and 5. Each line codes the meetings for one round.
Take this bunch of numbers and follow it by 8 lines for the contestants, and *numberlines* will put them in the best order.
Suppose one of the players is in a wheelchair, and you want them to be the pivot, then the first 4 numbers would become "7 4 2 7 1" so now there are two groups of size 7 and 1, and the program will renumber within the group from 1 to 7, but pair 8 will stay pair 8.

In a Mitchell movement you would have two groups of equal size. Remember that in a Mitchell with 22 pairs the NS pairs will be numbered 1-11, and the EW pairs 12-22. The program is very general and can be programmed for about all cases.

The program has been used with individual movements too. In an individual you either have 4 or 6 meetings per table per round, depending on if you count your partner. Especially with a big movement the number of numbers becomes staggering.

Output of numberlines

The program *numberlines* can take a while to run. It gives progress messages while it does so. The below example is from a run for a 72 pair endless Howell. Eventually it finds a best result with the last same-country meeting at round 44 of 71. The notation 46/7685 means the last round with same-country meeting is 46, and some sort of sum-of-square is 7685. If the latter number becomes lower, it means other same-country meetings before round 46 were closer to the beginning. Eventually the program did close to three million iterations to find this. Faster computers will do more iterations, the limit is clock time. Currently 15 minutes.

```
At try 1(51840) round 59/13006 beats 1000000/1000000
At try 2(51840) round 56/7540 beats 59/13006
….
At try 968(51840) round 46/6689 beats 46/7865
At try 5352(51840) round 45/8807 beats 46/6689
Extending search to 53520 tries
….
At try 408897(2045210) round 45/5739 beats 45/5795
Extending search to 2862279 tries
```

```
At try 476020(2862279) round 44/7786 beats 45/5739
Found best possibility at try 476020(2862279), score 44/7786
```

Auxiliary programs

There are a couple of auxiliary programs to generate movements. These were mainly used in testing this software, but they could prove useful.

- The program *mitchell* gets one command line argument, the number of tables. It then generates a complete Mitchell movement, including a skip when the number of tables is even.
- The program *baromhowell* gets one command line argument, the number of pairs. It then generates a complete barometer Howell movement, with the highest pair stationary at table 1.
- The program *hhj* reads movement data as also used by *numberlines*, but with only one round of meetings. It then completes the movement using the normal rules for movements out of the well-known big orange book. There are two groups of pairs, the first group moves, the second group is stationary.

Potential additions

As it is currently written, *numberlines* tries to get all meetings by contestants in the groups as early as possible, without regard to the group. Sometimes there are suggestions that contestants of a certain group must meet earlier than other groups. This could be accommodated with a slight alteration to *numberlines* but that alteration will only be done on request.

Wrapup

This set of programs can solve problems for tournament organizers. This is not easy stuff but can be very useful if you know what you are doing. I had fun writing this and will not charge for any use. I would consider it polite if anybody using this software would let me know what it is used for, and into what software they are integrating it.

Suggestions, problems, or other reactions can go to

Hans van Staveren <sater@xs4all.nl>