The Squeaky Wheel Algorithm:
Automatic Grouping of Students for Collaborative Projects

Steven L. Tanimoto
University of Washington
Dept. of Computer Science and Engineering
Box 352350, Seattle, WA 98195, USA
tanimoto@cs.washington.edu

1 Introduction

Collaborative learning is playing an increasingly important role, as schools work to make
education engaging for students and responsive to needs in the job market. Small groups offer
online learners opportunities to engage in discussions without having either full responsibility for
keeping a conversation going or the opportunity to drop out and be merely a lurker. In order for a
group of students to function effectively, it’s important that its members get along with one
another and have positive attitudes about each other. Effective groups are a prerequisite for group-
adapted pedagogy; thus group formation should be considered carefully before much effort is put
into tailoring lessons for the groups. In some tight-knit school communities, students all know
each other and may readily be able to form their own teams, but in larger classes, help may be
needed. In large university classes, if students are permitted to view each other’s home pages, they
can easily come up with suitable preference ratings for automatic grouping.

Here I present a particular method of automatic grouping called the “squeaky wheel algorithm”
and I discuss its advantages over more complex schemes. The tool has been used for grouping
students from classes of about 60 into teams of 4 students each. The method tends to satisfy
students, because of the straightforward manner in which their preferences are taken into account.
The algorithm is efficient, deterministic, and more understandable to students than algorithms such
as genetic optimization for the set partitioning problem, or methods based on detailed personality
models (Bekele, 2005), although the preference ratings of each student can be considered to be
part of a user model.

2 A New Algorithm

The algorithm uses two kinds of input information: a set of preferences for each of the students in
the class, and a list of the sizes of the groups to be filled. Before the grouping algorithm is run,
each student enters a set of preferences. If any students fail to enter preferences, then neutral
“default” values are used for those students. In the implementation of the algorithm described
here, a web server and browser are used by students to enter their preferences. Each student first
logs into a secure system (in this case, an online learning system at the University of Washington
called INFACT). He or she goes to the grouping-preferences web page and makes one rating for
each of his or her classmates. There are five choices for a rating: 4 is Strong positive, 3 is Mild
positive, 2 is Neutral, 1 is Mild negative, and 0 is Strong negative. This scale is fine enough so
that students can accurately state their preferences, and yet it does not impose an overly complex
task upon students. The collection of all this information for the class consists of a (two-
dimensional) matrix Pref, where Pref(i, j) is the value given by the i-th student for the j-th student.

Preference values are used in calculations of “mutual compatibility.” Two students are
mutually compatible provided that they each rate the other highly with their preferences. A
measure of mutual compatibility between two students is defined by the formula M(i, j) = Pref(i, j)
* Pref(j, i). By making this measure correspond to the product of the two preference values,
several desirable features are obtained. First, the value is symmetric. Second, either party to the
potential relationship has “veto power.” When forming groups, another measure of compatibility
is needed: one which helps compare a candidate for a group to all the members already in the
The individual-to-group compatibility $M_{ig}(i, G)$ is defined as the mean of the compatibility values between $i$ and all the current members of $G$. Suppose $G = \{n_1, n_2, \ldots, n_q\}$. Then,

$$M_{ig}(i, G) = \frac{1}{q} \sum_{j=1}^{q} M(i, j)$$

The Squeaky-Wheel algorithm begins by computing for each student $i$ the mean rating $\mu_i$ given by that student, as well as the standard deviation $\sigma_i$. The students are then sorted into order of decreasing $\sigma$. The first student in this list is the “squeakiest wheel,” because he or she has entered the most highly varying ratings of the other students. This student is denoted $sw(1)$, and the others are $sw(2), \ldots, sw(n)$. At each stage of the algorithm, there is a current group $G_h$ to be filled, and there is a current version $SW_i$ of the list of squeaky wheels remaining to be put in groups. The iteration starts with $h = 1$, and $G_h = \{\}$. Also, $i = 1$, and $SW_i$ is set to be the entire list of $n$ students, in the order of decreasing $\sigma$.

Now the following “outer loop” is repeated for $h = 1, 2, \ldots, n_g$.

1. $G_h$ is set to $\{\}$.
2. The squeakiest remaining wheel (the first element of $SW_i$ ---call it $w$) is removed from $SW_i$ and placed into $G_h$. So now, $G_h = \{w\}$, and $SW_{i-1} = SW_i - \{w\}$. Also, $i \leftarrow i + 1$.
3. Next, the following “inner loop” is repeated for $k = 2, 3, \ldots, \text{Size}(h)$.
   a. For each of the remaining students, $q$, compute $M_g(G_h, q)$, and let $q_{\text{min}}$ be the first one having the highest value. Remove $q_{\text{min}}$ from $SW_i$ and put it into $G_h$. Then set $i \leftarrow i + 1$.

The group $G_h$ has now been filled so go on to the next value of $h$.

3 Implementation and Usage Experience

The Squeaky Wheel grouping algorithm was coded in Common Lisp as a CGI program within the INFACF online learning environment (Tanimoto et al, 2000). The algorithm is efficient in comparison with algorithms that find optimal partitions of a set or of the nodes of a graph (Balas and Padberg, 1976). Such a problem is, in the general case, NP-complete, or practically intractable for all but the smallest instances of the problem. Algorithms that find good suboptimal solutions exist (Greene, 2001), but those algorithms contain nondeterministic (random) elements meaning that it may be difficult to get repeatable results, they are far less efficient, and they may be difficult to explain to students.

The algorithm has been used in two medium-sized classes given at the University of Washington. One of them was a senior-level course on artificial intelligence with 58 students, in which many of the students knew each other, but many knew only few other students in the class. The other class was a 60-student course on programming languages, also with a mix of prior student-student acquaintances. In each class, the groupings produced by the algorithm were highly successful in terms of student satisfaction. In both courses, there were no complaints about the group composition.

Acknowledgments. This project has been supported in part by the National Science Foundation under Grants EIA-0121345 and 0537322. Thanks to R. Ladner and R. Rice for comments on the algorithm.

References