

Volunteers in Large Clubs: The Theorist's View

John Mount*

February 26, 2009

1 Introduction

A recurring problem in large clubs and institutions is: how to recruit volunteers for various club projects. As a club gets larger it becomes progressively more difficult for the right groups of people to find each other and coordinate volunteer activities. This loss of effectiveness is often (incorrectly) perceived as lack of interest or uncharitable attitudes among club members. However, the root cause is actually a mathematical problem brought about by the fact that: as a club grows the number of possible pairs of people grows much faster than the number of people.

This problem presents a good opportunity to demonstrate “how a theoretical computer scientist thinks” to a general audience. I will work through some interesting aspects of the problem and touch on some ideas that have been used to solve this problem. This write-up isn't short but it is intended to be an easy to read walk through (with no math) of how a theoretical computer scientist thinks about this kind of problem.

2 Aside: What is a Theoretical Computer Scientist?

Essentially a theoretical computer scientist is a type of mathematician. Theorists (as they are called) use mathematical techniques to study very simple procedures. The mathematics is often very difficult because even simple procedures can have incredibly complex consequences in the long run.

For example: one masterpiece of theoretical computer science is Donald Knuth and Arne Jonassen's analysis of a procedure for maintaining a sorted list of just three items.[8] Notice there is no mention of computers or even of mathematics in the problem: just a keeping a list of three items in order. Anyone can keep a list sorted as we ask them to add and remove items- and for just 3 items the procedure is so simple as to be called trivial. However, Knuth and Jonassen required 21 pages of mathematics to precisely calculate how much work is needed to keep the list in order. In addition they were able to convincingly argue that no simpler analysis could find the right answer.

*<http://www.mzlabs.com/>

This incredible difference of scale in problem complexity and solution complexity is one of the hallmarks of theoretical computer science. It is also why deep down theorists value simplicity so highly: they know how quickly complexity drives up expense.

3 Back To The Problem

Pretend you are running a growing club or volunteer organization. You wish to allow volunteers to form small groups and perform charitable works. You start with two methods to organize volunteers: announcements at the club and having organizers ask individuals to join their groups. For quite a while this works well.

As the club gets larger you expect the amount of service the club can provide to get larger (you have more people, so you have a lot more capacity to do good). This means more small groups. Soon allowing calls for volunteers at your meetings is eating up too much time and organizers have a harder and harder time finding volunteers.

What has happened? Has the club lost its spirit? Are the members becoming uncharitable. No. What has happened is that logistics and communication get progressively more difficult as clubs get larger because as the club grows the number of pairs of people grows much faster than the actual number of people. As the club forms more groups the groups tend to be specialized so each group/task organizer ends up asking more and more of the wrong people (people who would be better suited to another volunteer task) to find matches.

4 Various Solution

The solution is more organization. But what kind? It is often a surprise which type of solution is best so the theorist usually explores and rejects a large number of different solutions before settling on a method.

Since I am trying to show the theorists way of thinking about this problem I will list a few different methods of organization. The job of the theorist is to have a large ready set of analogous processes and to see if a given problem can be re-cast into one that has a known solution.

4.1 Hierarchy

Companies typically organize in a hierarchy with employees sorted by type of activity. If a company wants to add a task they know fairly precisely which subset of their employees to assign it to.

For volunteer clubs there is usually some hierarchy- but too rigid a structure is undesirable.

4.2 Scrip

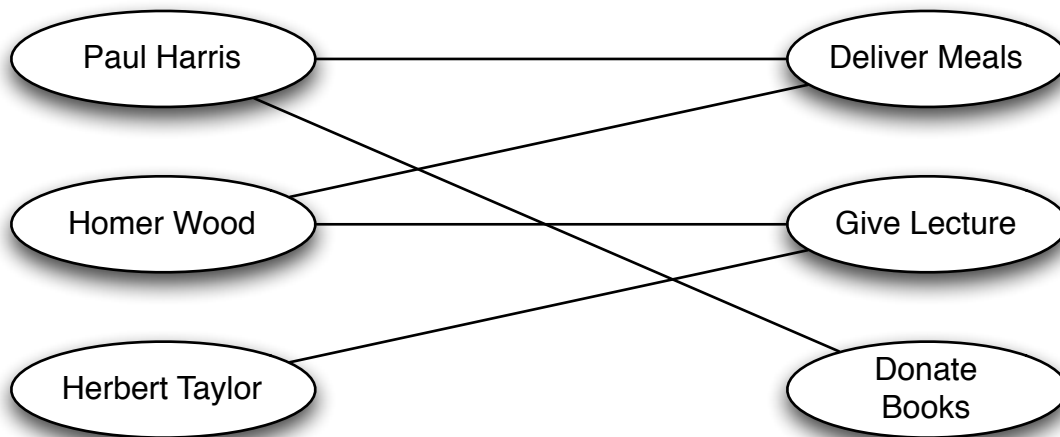
Joan and Richard Sweeney described the dynamics of a baby-sitting coop that used scrip (any substitute for currency)[11] as its organizing tool. The idea was: by exchanging coupons (or

scrip) families could recruit each other as volunteers to babysit for each other. The fascinating thing is that this economic style solution quickly developed all of the complexities of an actual economy (inflation, deflation, currency policy and business cycles). Similar problems could be expected with an auction (or more fashionably a “mechanism design” [10]) based approach.

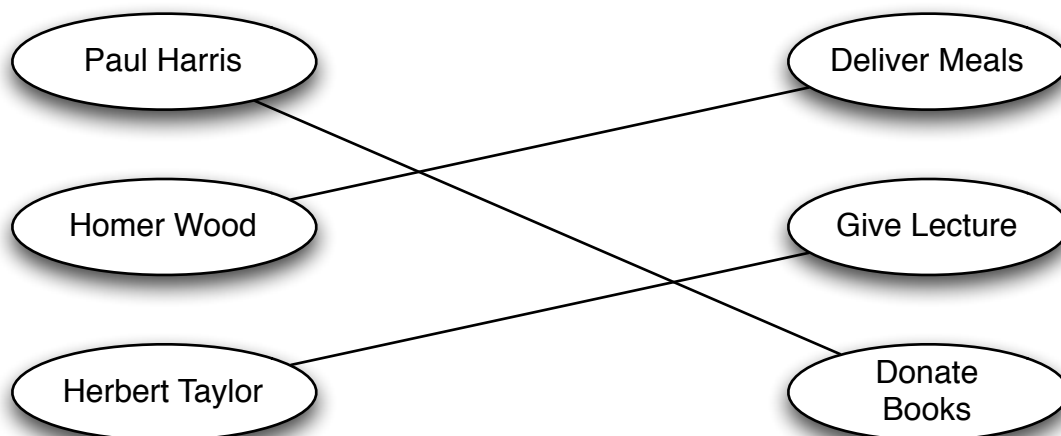
Computer scientists have a catchy phrase for expected outcome when you dogmatically apply a method to solve a problem: “now you have two problems” [4].

4.3 Matching Theory

Matching Theory is the idea of organizing the information into two lists: one of potential volunteers and one of tasks. Typically one list is written in a column on the left and the other is a column on the right. From each potential volunteer we draw a line to each and every task they are willing to take on. For example we could have the following three volunteers and tasks:



A “matching” is when we remove many of the lines and leave only the assignments. For example the previous diagram of possibilities would allow us to make a complete assignment as given below:



In matching theory any assignment or recruitment of volunteers is called “a matching.” There are various ways to attempt to build a matching.

4.3.1 Random Matching

One method would be to randomly pick pairings that are allowed by the original diagram. The main benefit is that by centralizing the book keeping we have allowed people with more limited availability (in this case Herbert Taylor) to enter their preferences without a lot of communication. However these harder to match people (that are available for fewer tasks) can easily be missed in a random matching. For example Herbert Taylor can only be matched to his task as long as Homer Wood has not already taken it. A very good study of how effective random assignments are is found in [7].

4.3.2 Greedy Matching

Another method is the so-called “greedy method.” A greedy method scans the list and (with a very myopic view) picks the best match. For example we could design our greedy method to proceed across the names in order and match each person up with the task they point to that has the fewest remaining edges. We would first match Paul Harris to “Donate Books”, then Homer Wood to “Deliver Meals” (as this now has only one edge remaining after Paul Harris is removed) and finally Herbert Taylor to “Give Lecture”. The method is not always guaranteed to produce a best matching but a very good analysis of how well it does work found in [3].

4.3.3 On-line Matching

The Karp, Vazirani and Vazirani paper (“An Optimal Algorithm for On-line Bipartite Matching”, [7]) actually proposes an algorithm that combines many of the ideas we have already mentioned. The authors noticed that the worst thing about the random matching is that it tends to match people who are easy to match too early (instead of holding onto them for later). So instead of matching at random the new algorithm first places all of the

potential volunteers into a list in random order. It then inspects the tasks one at a time and always assigns the available volunteer that is highest on the volunteer list. What is going on is that the algorithm is building a preference for matching those that are hard to match early. When we examine a task and see that a volunteer very high on the list is available it means that they must not have been available for very many of the tasks we previously examined (else they would have been already matched). So it is a good idea to match them while we can.¹

This idea is so powerful that it has been suggested as an improvement to the bidding model used to price Google AdWords[9].

4.3.4 Stable Marriage

Stable marriage theory[5] is an idea that looks at the entire diagram at once and allows us to assign preferences to each matching. In its most general form each potential volunteer submits a list of tasks they are willing to perform and orders them by their preference. Each task organizer also submits a list of volunteers they are willing to accept ordered by their own preferences. The stable marriage algorithm then finds a very special matching called a stable marriage. This matching tends to be very good. In fact this is the algorithm used to assign interns to hospitals.

A stable marriage is an assignment of pairs (a matching) where no assignment swaps are practical. That is if we pick a volunteer and a task the volunteer is not matched to then either the volunteer already has an assignment they like more than the task or the task already has an assignment they like more than the volunteer. Thus there is nobody who wants to trade tasks that can find a task willing to have them.

Gale and Shapley proved a stable marriage always exists and gave an effective algorithm for finding one. However this algorithm is not well suited for situations there are many tasks volunteers are unwilling to do. Notice, however, what ideas stable marriage shares with the on-line matching procedure (such as use of pre-prepared sorted lists).

4.3.5 Max-Flow

Another “look at the whole diagram” idea is called “maximum-flow”[2]. In this case each line is considered as a directional pipe able to move 1 unit of fluid per hour from the left to the right and each volunteer is a source of 1 unit of fluid per hour and each task is a drain with a capacity of 1 unit of fluid per hour. The maximum flow algorithm can find a minimal subset of pipes that carry as much flow as possible. This minimal configuration is in fact a matching that assigns all tasks and volunteers (as in our second diagram, such a matching is called “maximal”).

The maximum flow algorithm can include preference weights (allow some respect of volunteer preferences in addition to allowing each volunteer to mark a subset of tasks they

¹That is the intuition. As is often the case in theoretical computer science the intuition is in fact too hard to work with and the proof that the technique is good has to proceed on a longer and more difficult path. Also it is typical of theoretical computer science that the algorithm being analysed is much simpler than a number of obviously better heuristics. The heuristics may be better, but any improvement that yields too much complexity interferes with analysis.

are willing to take). This is a case where the details of how maximum flows are computed is irrelevant, we just need to remember it is possible and see how to encode our problem as a flow.

4.4 The Liebermann Queue

The graduate students of Carnegie Mellon's School of Computer Science have, for over 25 years, used a system called "the Liebermann Queue" (named after its inventor Bob Liebermann)[6] to organize and encourage volunteerism.

The principles are: we take volunteerism as a responsibility and getting the task you want as a mere privilege. The queue is just a sorted list of the graduate students. A second list of tasks sorted by when they are needed is also maintained. Both lists are publicly available. Anybody can volunteer for any task at any time. When you complete a task your name is moved to the bottom of the queue (causing the the people you pass to each move up one position in the queue). The final point is that if you reach the top of the queue (which happens if you volunteer for tasks a significantly slower rate than your peers) the queue manager (originally Bob Liebermann) could forcibly select you for a task. So it is in your interest to periodically inspect the available task list and see if there was one you would like to do (to stay away from the top of the queue).

You can see this system cuts down immensely on the required amount of communication. Each member only needs to check the public task list every once in a while and the queue manager is always either accepting volunteers or forcibly assigning tasks (so there are no "declines").

It must be admitted this system is fairly radical. First the system is very normative (imputes ethical judgements on actions) and is in fact a (benevolent) shame culture. This may not be appropriate for many organizations and may not be appropriate when the task frequency is high.

5 Conclusion

There are a number of ways of improving the quality of volunteerism in a club. Most of the ones I explored involve some form of central tracking of volunteers and tasks and replace inefficient direct communication with sorting performed on the abstract records.

A number of important research fields can have their spirit summed-up in a single sentence. Rudolf Beran said: "statistics is the study of algorithms for data analysis"[1]. Stanislaw Ulam said "the best mathematicians see analogies between analogies." To this I would like to add "theorists build analogies between processes."

References

- [1] BERAN, R. The impact of the bootstrap on statistical algorithms and theory. *Statistical Science* 18 (2003), 175–184.

- [2] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., AND STEIN, C. *Introduction to Algorithms, 2nd edition*. MIT Press, McGraw-Hill Book Company, 2000.
- [3] DYER, M., FRIEZE, A. M., AND PITTEL, B. G. The average performance of the greedy matching algorithm. *The Annals of Applied Probability* 3, 2 (1993), 526–552.
- [4] FRIEDL, J. Source of the famous now you have two problems quote. <http://regex.info/blog/2006-09-15/247#comment-3085>.
- [5] GALE, D., AND SHAPLEY, L. S. College admissions and the stability of marriage. *The American Mathematical Monthly* 69, 1 (Jan 1962), 9–15.
- [6] HANCOCK, J. Vasc queue description. http://vasc.ri.cmu.edu/old_help/Admin/Queue/descrip.html.
- [7] KARP, R. M., VAZIRANI, U. V., AND VAZIRANI, V. V. An optimal algorithm for on-line bipartite matching. *STOC 22* (1990), 352–358.
- [8] KNUTH, D. E., AND JONASSEN, A. T. A trivial algorithm whose analysis isn't. *Journal of Computer and System Sciences* 16 (1978), 301–322.
- [9] MEHTA, A., SABERI, A., VAZIRANI, U. V., AND VAZIRANI, V. V. Adwords and generalized on-line matching. *Journal of the ACM* 54, 5 (2007).
- [10] NISAN, N. Introduction to mechanism design (for computer scientists). (*Book*) *Algorithmic Game Theory* (2007).
- [11] SWEENEY, J., AND SWEENEY, R. J. Monetary theory and the great capitol hill baby sitting co-op crisis. *Journal of Money, Credit and Banking* 9, 1 (Feb 1977), 86–89.