In Praise of Tweaking: A Wiki-like Programming Contest

By Ned Gulley, The MathWorks, Inc.

I know something you want to know. But will I tell you? Why should I? What's in it for me? Internet technology makes the simple act of online collaboration easy, but human nature stubbornly resists change. Why should I collaborate with you? What would convince me to share my ideas with you, given that you might use them to compete with me? In a larger sense, how might we build a system that encourages people to collaborate in, for example, writing useful code? These are some of the questions we have explored with a new kind of open source programming contest.

Programming contests have become a regular feature of geek culture. The ACM, for example, runs a collegiate programming competition that has grown to include almost 4000 teams from around the world [1]. Contests may bring all the participants to one location, or they may happen across the Internet, but they tend to share a common format: given a specific problem, you have a limited amount of time to write better code than anyone else. This kind of contest is good for determining the single most talented individual from among a group of contestants, but it has a decidedly academic feel. What if there were a contest that more accurately modeled the way ideas really move through the world? Suppose, once an idea had been put forward by one person, it could then be freely adopted and modified by anyone else even as the contest is still running? The winning entry for this kind of contest would be an amalgamated effort by many people, people who were simultaneously competing and collaborating. This approach is more like the messy, organic way in which much software, particularly open source software, actually gets built. Presumably, then, an open source programming contest might show us something about how innovation works in the real world. For several years, we have been running exactly this kind of contest using the MATLAB programming language, and the results have given us a fascinating quantitative perspective on the dynamics of innovation and reward in collaborative programming [2]. Over time, we have observed a striking resemblance between our open source contest and wiki-based web sites. A wiki is a bare-bones document management tool for online collaboration. Very simply, it is a page or a collection of pages that can be modified by anyone viewing it. This simple rule has profound consequences. By drastically lowering the cost of participation, the number of participants is correspondingly expanded. Surprisingly, documents created and maintained on wikis are often cogent, helpful, and well-maintained. Enormous projects have been built as wikis, the most spectacular of which is the Wikipedia, an entire encyclopedia which spontaneously grows in volume and value through wiki-induced collaboration [3]. Our contests resemble a wiki in the sense that anyone can modify any of the code on display. As with wikis, the result is a fertile meeting of the minds, and a model for successful collaborative design.

How the contest works

The MATLAB language (developed by The MathWorks, Inc.) is optimized for high-speed number crunching. It is particularly useful for the rapid prototyping of algorithms. Longtime practitioners of the language develop tricks and techniques that trade off speed of implementation, speed of execution, elegance, and compactness. We ran our first contest five years ago in the hope that it would be an entertaining way to encourage MATLAB users to show off and to share their skills. Its success convinced us to make it a regular event. We try to hold contests twice a year with each one lasting roughly a week. Since January 1999, we have run eight different contests. These contests typically take the form of trying to solve a single difficult optimization problem in the least amount of time. The traveling salesman problem is the canonical example of this: what is the shortest possible round trip a salesman can make through a given list of cities? Contestants must write MATLAB code that, given the location of all cities to be visited, returns an ordered list of how best to visit these cities. They never get to see our test suite. They simply submit their best algorithm, and the computer grades it and notes how long it took to run. Thus their algorithm has a result (distance traveled, which is to be minimized) and a CPU time (also to be minimized). We combine these two into a single score, using an exponential penalty for CPU time as shown below.

$$score = k_1 \cdot distance + k_2 \cdot e^{k_3 t}$$

The values k1, k2, and k3 must be tuned for each contest. If we don't penalize CPU time at all, the entries may take too long and bog down our system. On the other hand, if we penalize CPU time too much, speedy but boring algorithms result. Problems are chosen so that no one has enough time to find the absolute best solution. Contestants must explore the landscape of tradeoffs between algorithm performance and execution speed. This format is important, because it means that there is always room for improvement as the week progresses.

From a contestant's point of view the contest consists of three primary web pages: the current standings, a page for viewing the code behind any of the entries, and a page for submitting a new entry. As noted, the unusual feature of this contest is that contestants submit code that is immediately scored, ranked, and displayed for all to see. In fact, as with the "Edit this Page" button on a wiki page, the contest is specifically designed to encourage participants to steal each other's code. As shown in Figure 1, one straightforward process lets you view the code for any entry (typically the leading entry), make a change to it, and submit it with your name on it.

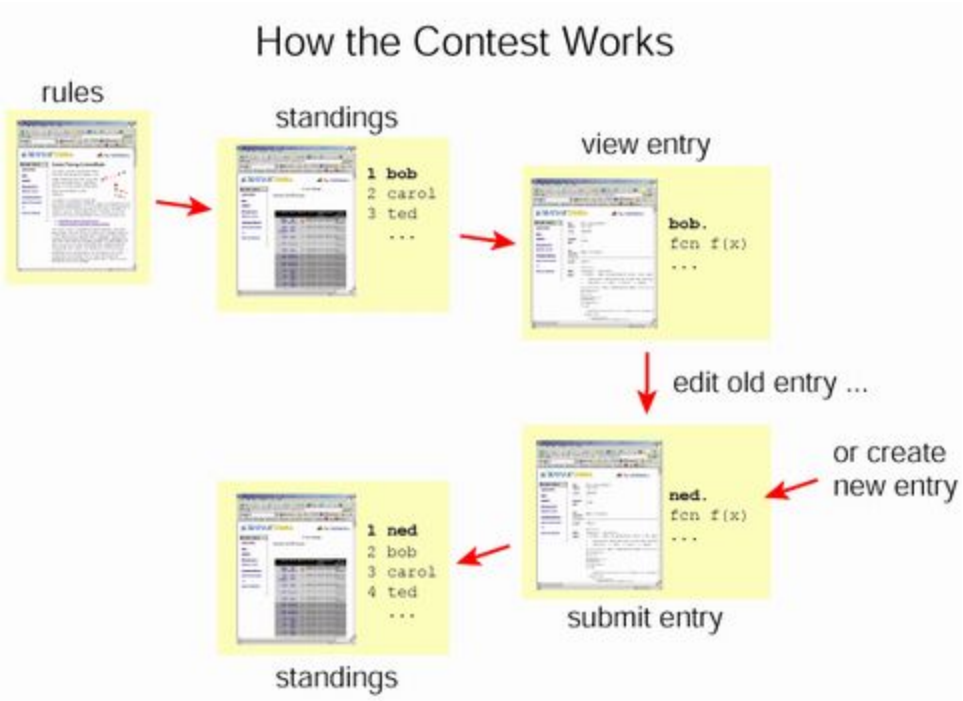

Figure 1. How the Contest Works

Suppose, for example, you see the leading entry includes the following line.

```
x = [1 2 3 4 5 6 7 8 9 10];
```

As a MATLAB expert, you happen to know that the line

```
x = 1:10;
```

has the same effect and runs much faster. You only need to push the "Edit this entry" button, make your change, and then submit it to the scoring queue. With luck, you will jump into first place.

Contest examples

In addition to our own variant on the famous traveling salesman problem, here are some other examples of contests we have run.

- Mapping Mars [1647 entries] Given several robotic rovers and a rough map of part of the surface of Mars, find optimal routing so as to explore the most area in the least time.
- Protein folding [2437 entries] A protein is a string of amino acids, each of which is either hydrophobic or hydrophilic. Given such a protein, fold it in two dimensions so as to

minimize the distance between hydrophobic amino acids.

- Mastermind [1138 entries] Solve for the hidden colored pegs in a generalized Mastermind problem that may have any number of pegs and colors.
- Molecular Modeling [1631 entries] Given a table of distances between various pairs of atoms in a molecule, reconstruct the molecule's shape.

There are generally on the order of 100-150 contestants, although this number is hard to pin down since people don't always consistently report their names. Some contestants choose to submit one or two entries, but others enter dozens or in some cases literally hundreds of algorithms, improving them steadily over a period of days.

Visualizing the results

To understand the nature of the contest, it's helpful to visualize the results. Figure 2 shows the 977 passing entries from our Molecular Modeling contest [4]. Every blue dot is a different entry. We are plotting the score of every entry (lower is better) against the time at which it was submitted, so the horizontal axis spans one week. The winning entry is defined as the entry with the lowest score when the contest closes. The red line running along the bottom traces the current best score at any point. Since the leading entry is always the one with the lowest (best) score, the line gets steadily lower until it reaches the winning entry in the lower right corner.

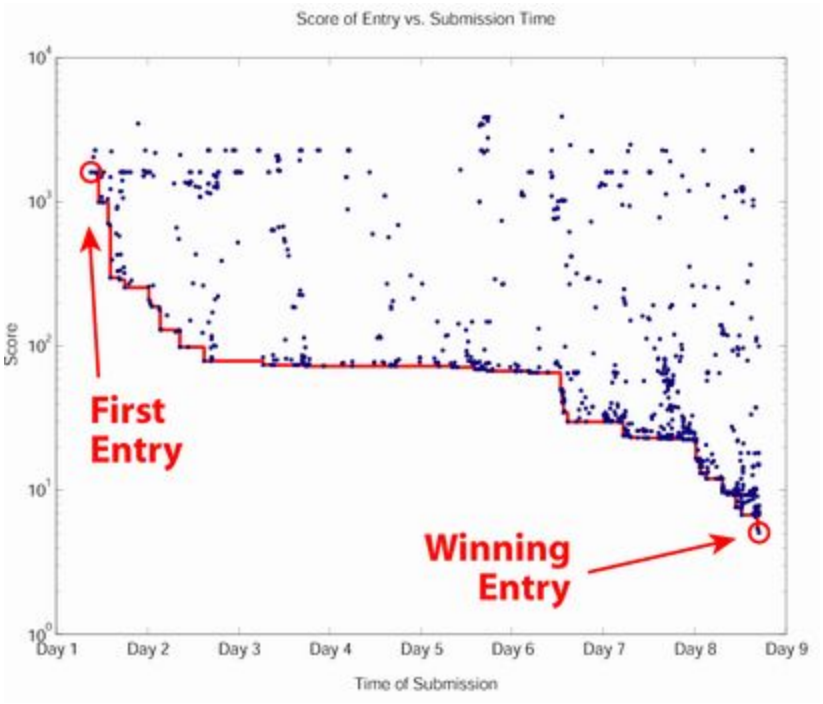

Figure 2. Score of Entry vs. Submission Time

Where the red line is basically flat, the improvements being made to the code are small. Big vertical drops are rarer and generally indicate fundamental improvements to the algorithm. A leading strategy tends to be modified over and over before a significantly different approach displaces it. The interplay between these two approaches leads to the stair step pattern seen in Figure 2.

We have some extra information about these entries that is not displayed in Figure 2: we know when one entry has been derived from another entry. Thus if an entry named, for example, "TurboSolver" inspires someone else to modify it and call it "Son of TurboSolver", we can show this relationship with a line between the two entries. This descent with modification can lead to complex family trees. The next plot, Figure 3, shows the same contest data as before with lines connecting all related entries; for clarity we have highlighted one cluster of related entries in red. The red dots connected by red lines are entries that all belong to a single family of entries. The leftmost entry inside the black circle is the progenitor of the family group. It stands apart from all preceding entries in the sense that the author did not credit any other entry as its "parent." This generally means the algorithm is new, although the author may have chosen to obscure its inspiration. By following families of entries and modifications to the code, we can see how and when improvements are introduced.

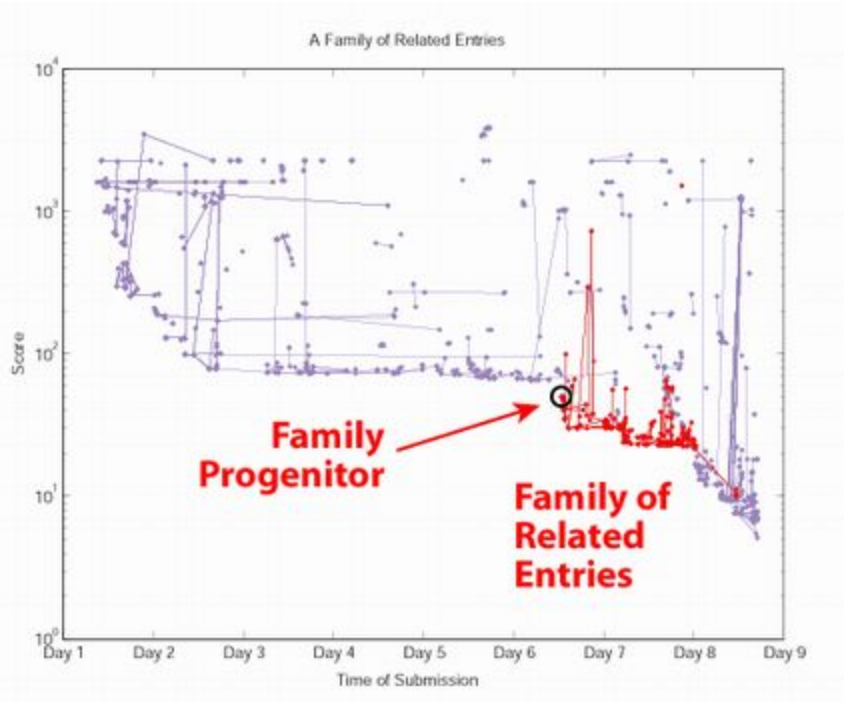

Figure 3. A Family of Related Entries

Figure 4 shows how many lines of code each of the 90 leading entries in the Molecular Modeling contest contained. To get a sense of how much code is conserved over time, each bar

shows in green how many lines of code are identical to lines in the previous leader. The red cap to each bar shows how many lines deviate from the previous leader. Clearly code is highly conserved during the contest, often to the extent that only a single line changes from one entry to the next. You can also see that while the length of the code tends to grow over time, it can decrease too as unnecessary code is chopped away.

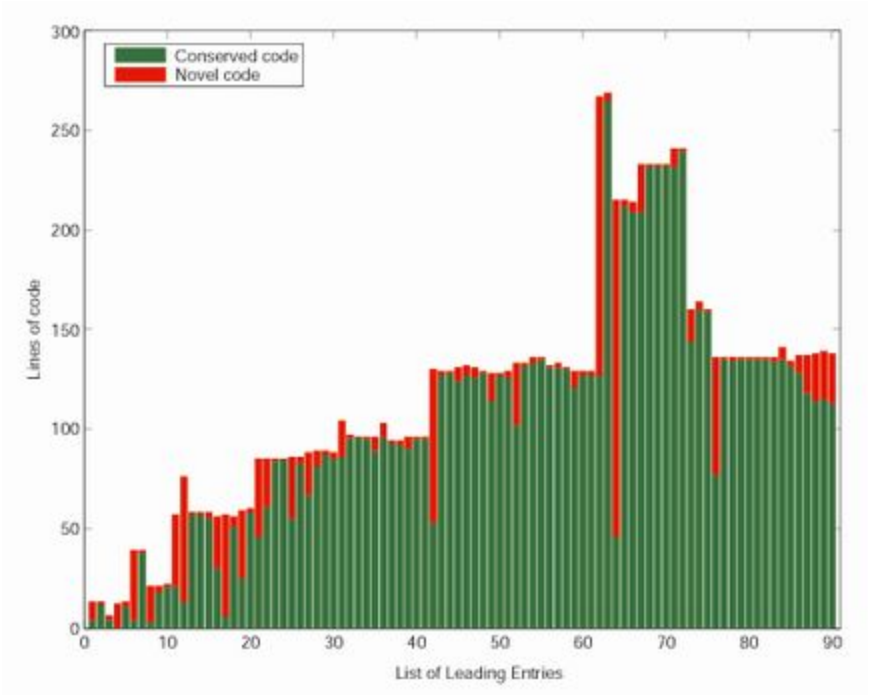

Figure 4. Code Conservation in the Leading Entries

Patterns of participation: Tweaks and Leaps

Broadly speaking, you have two choices if you want to join the contest. You can submit a completely new approach in the hopes that it will represent a leap in performance, or you can modify an existing program. We use the word "tweak" to describe the process of making a minor modification to someone else's code. The contest is characterized by a sort of zigzag between leaps and tweaks. The leader is constantly being probed for weak spots, thereby pulling any slack out of the algorithm. Long stretches of tweaking battles can be suddenly punctuated by dramatic shifts in the code. When one of these big shifts occurs, it also opens up fresh opportunities for tweaking, and swarms of curious competitors descend upon and begin tightening up the new leader.

The nice thing about tweaking is that it dramatically lowers the barrier to entry. You don't have to understand the algorithm involved; you need only know that you are replacing a slower line with

a faster line that does the same thing. No improvement is too small to be worthy of consideration. If you are the first person to notice the lead entry declares a variable that never gets used, you can simply delete the offending line and resubmit the file with your name on it and you will (momentarily, at least) be vaulted into first place. Initially most people find tweaking distasteful or even parasitic, particularly those who have been tweaked out of first place. But tweaking turns out to be the fuel that drives the entire contest. It offers an immediate reward to the tweaker --- for an investment of a few minutes, your name appears on the leader board, wreathed in glory. The practice is also a call to arms for the original author ("How dare someone tweak my code!"). If you get tweaked, you want to know about it. And you may work very hard to tweak your way back into first place.

This situation is very analogous to someone who fixes a spelling error in a Wikipedia article. There can be a genuine thrill as you realize your small change has "gone live" to be seen by the entire world. This low level activity, whether tweaking code in a contest or spellchecking wikis, provides an addictive gradient of reward that pulls people into higher levels of participation. We find that tweaking is the thing our contestants most often complain about, and at the same time it is the feature that keeps them coming back for more. Our discussion boards swirl with questions like this:

- Who deserves the most credit for this code?
- Who is a big contributor and who is "just a tweaker"?
- What is the difference between a significant change and a tweak?

These kinds of questions bedevil real-world software projects. There seems to be a cultural predisposition to find and glorify the (often mythical) breakthroughs of a lone genius. Since this model doesn't always match reality, these questions don't have satisfying answers. Happily, the contest framework acts as a solvent that minimizes this kind of I-did-more-than-you-did bickering and maximizes fruitful collaboration among many parties. Wikis share this property. It has been noted that, whereas newsgroups and weblogs tend to intensify flame wars, wiki documents soothe them [5]. In the case of our contest, bickering still happens, but by and large, it's clear everyone is having a good time. Part of this successful formula is the fact that we don't offer valuable prizes to the winners of our contest. The primary reward is social. Again, by way of analogy, suppose Wikipedia contributors were paid large sums of money based on how many of their words persisted in the articles they touched. You can imagine the noise that would result. An enterprise held together by reputation is easily damaged by cash.

Conclusions

On first hearing how the Wikipedia site works, people are often scornful, incredulous, or simply dismissive. It can't possibly work. How could it? Similarly, the MATLAB online programming contest is built upon an almost paradoxical premise: that a contest can be collaborative. Against all expectation, the back and forth drama of leaps and tweaks turns MATLAB programming into an entertaining spectator sport. Again and again contestants tell us how much fun they're

having. "It's one of the most addictive and compulsive things I have tried," says one. "Forgive me, for I have tweaked… I resolve to tweak no more, but I am not sure I can stop myself," says another. And this: "At home, although I am a father of three children, my full time job was working on the contest." As with wikis, the success stems from the ego reward of immediate gratification compounded by a sense of participation in a large and worthwhile activity. The resulting high level of participation creates an atmosphere in which supercharged algorithmic evolution takes place. The winning entry in each contest embodies the tangled effort of dozens of people.

To return to our earlier question: why should I collaborate with you? The answer is that I value your opinion. Since your opinion of me depends on the value of the ideas I put into play, I will work hard to give you exactly what you want. Good collaboration environments continuously and generously reward sharing with reputation-enhancing praise. The successful push-pull of collaboration and competition in our contests echoes the popularity of both open source programming and wikis and is certain to find its way into many enterprises as collaborative design becomes commonplace.

References

1. ACM International Collegiate Programming Contest Hall of Champions,http://icpc.baylor.edu/past/default.htm
2. Gulley, Ned, Patterns of Innovation: A Web-based MATLAB Programming Contest, Extended Abstracts of CHI 2001, p. 337-338, 31 Mar-5 Apr 2001, Seattle, WA. ACM Press.
3. Wikipedia, http://wikipedia.org/
4. Molecule contest final report, http://www.mathworks.com/contest/molecule.cgi/final.html
5. Shirky, Clay, RSS, Echo, Wikis, and Personality Wars,http://www.corante.com/many/20030701.shtml#42346

Appendix

Visit the Contest Hall of Fame.
Here is some information about the May 2005 contest: Ants Contest Evolution.
Return to the Rambles weblog at starchamber.com