Patterns of Innovation: A Web-based MATLAB Programming Contest

Ned Gulley
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760 USA
+1 508 647 7331
gulley@mathworks.com

ABSTRACT

In this paper, we describe an innovative web-based MATLAB programming contest and point out some interesting connections between the contest and open source software development.

Keywords

MATLAB, programming contest, open source, evolutionary coding, collaborative development

INTRODUCTION

MATLAB, developed by The MathWorks, Inc., is a matrix-based language optimized for fast numeric computation. Because of its matrix-based approach, it is possible, using MATLAB, to write compact code that is nevertheless very expressive. MATLAB lends itself to rapid prototyping of algorithms, and longtime practitioners of the language develop tricks and techniques that trade off speed of implementation, speed of execution, elegance, and compactness. A contest is an entertaining way to encourage these programmers to both show off and share their skills.

Over the years, we have run several e-mail based MATLAB programming contests, in which a challenge was issued and anyone was welcome to send in an entry by e-mail to be tested against all other entries. These competitions were popular, but they were slow and tedious to score, resulting in a turnaround time on the order of a month or more. Looking to speed things up, we created a web-based MATLAB programming contest in which contestants submit code that is scored and ranked in real time [1]. Our primary goal was to provide an entertaining diversion to the community of MATLAB users while encouraging the exchange of good programming practices. The results have been satisfying: the on-line programming contests have been crowd-pleasing successes.

We have by now run three open contests as well as several internal to The MathWorks. Each lasts one or two weeks. Here are some important features of the contest.

- entries are automatically and immediately scored, ranked, and displayed
- the code, author, and score for all entries are visible to all contestants at all times
- anyone can modify an existing entry and resubmit it as their own (though the pedigree is tracked)

These contests generated a great deal of activity. Some contestants chose to submit one or two entries, but others entered literally dozens of algorithms, improving them steadily over a period of days. Interestingly, the leading entries represented the combined efforts of numerous contestants.

During the course of several competitions, we have gathered images and stories about how people who have never met are motivated to collaborate in writing highly optimized code. We had fortuitously developed a nicely instrumented open-source laboratory for observing the innovation at work.

HOW THE CONTEST WORKS

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, whether based on a previous entry or not. As they are submitted, entries are time-stamped and scored. Every entry has a name, an author, a time stamp, a CPU runtime, a metric of the algorithm's performance, and an overall score. Any entrant can find out within a few minutes if he has jumped to the top of the standings.

In order to rank the entries, we need to score them. There are two quantitative results for each entry: the performance metric or "goodness" of the result, and the speed with which it was computed. For each contest, we combined these two to make a single final overall score. The algorithm that calculates the final score must be tailored to each contest. If we don't penalize CPU time at all, the entries may take too long and time out, and the urge to streamline the code for speed will be minimal. On the other hand, if we penalize CPU time too much, speedy but boring algorithms result.

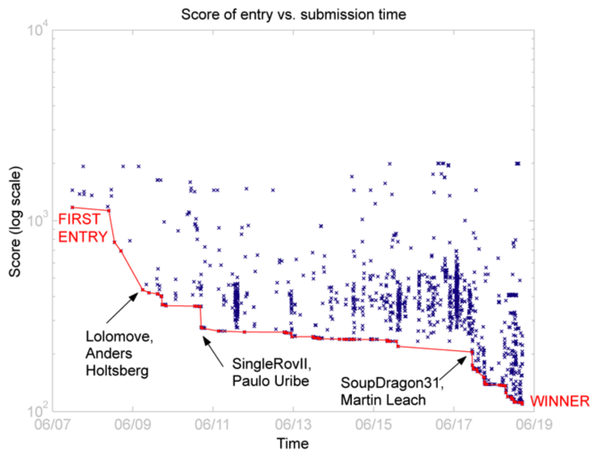
Contest examples

Here are the contests we have run.

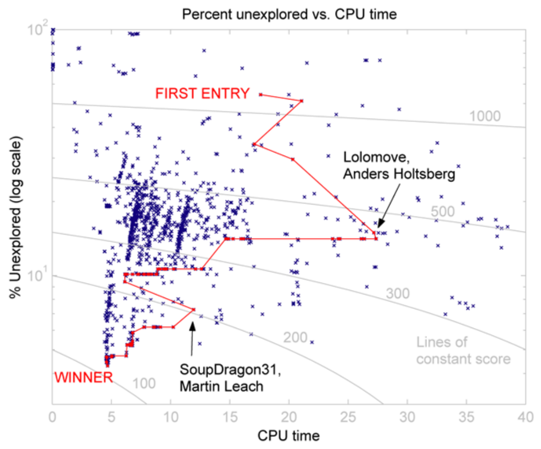
- Calculating a Fibonacci number [internal contest] Find the nth Fibonacci number as quickly as possible.
- Bin packing [1455 total entries] Given a large possible play list, put songs on a CD that will as nearly as possible fill up (but not exceed) the capacity of the CD.
- Optimal mapping [1647 total entries] Given a map of part of the surface of Mars and several robotic rovers, plan a route for the rovers that surveys the most area.

Visualizing the results

We used two plots to interpret the contest results. The first is a scatter plot with the submission time forming the x-axis and the score forming the y-axis. These results are from the Mars rover contest; every point is a different entry.



The line running along the bottom represents the current best score at any point. Since the leading entry is always the one with the lowest score, the line decreases monotonically toward the eventual winner on the far right.



This second plot shows performance metric on the y axis and CPU time on the x axis (in both cases lower is better). The line again traces through the leaders throughout the contest, moving from a high score in the upper center of the plot to the final lowest (winning) score in the lower left.

Interpreting the results: the zigzag of innovation

These diagrams let us visualize the evolution of the algorithms as the contest progresses. The contestants tried a diverse set of approaches in each contest. For example, generating Fibonacci numbers is not difficult, but we were surprised to discover eleven distinct strategies for performing the calculation. Broadly speaking, new entries were either incremental improvements (tweaks) or dramatic changes to the algorithm (leaps). In the second plot above, the line zigzags between these leaps and tweaks. Horizontal motion from right to left indicates speed improvements that do not improve the basic algorithmic performance. These are the tweaks. Big vertical drops indicate fundamental improvements in the algorithm.

Throughout the contest, the code in the lead position is constantly being modified by competitors in search of weak points in the code. If you can make the code even the tiniest bit faster, you become the leader. Tweaking the leader is tempting, because you don't necessarily 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. Tweaking may sound cheap or predatory, but even the most mindless tweaking can accelerate code over a surprisingly short period of time as the slack is pulled out of an algorithm. It has an appealingly egalitarian effect: no optimization is too small to be worthy of consideration.

Breakthroughs, or leaps in performance, are much rarer and describe a trajectory distinct from speed tuning. A leading strategy tends to be tweaked over and over, but not drastically modified for some time before a significantly different approach displaces it. The interplay between these two approaches leads to the zigzag pattern shown above.

CONCLUSIONS

The programming contest achieved a remarkable result: it turned MATLAB coding into an entertaining spectator sport. Feedback from participants was enthusiastic, and the contest led to many discussions about the relative merits of various coding techniques. We believe the contest was successful because it was

- competitive *(contestants are motivated)*
- real-time *(contestants remain engaged)*
- personal *(names are visible, discussion is encouraged)*
- open-source *(all code is visible at all times)*

Crucial to the appeal is the fact that you can quickly modify and resubmit someone else's entry. The winning entry in each contest represented the efforts of many people. Indeed, it's fair to say that no single person could have written such an optimized algorithm. This push-pull of collaboration and competition is strangely compelling, and it echoes the popularity of open source programming, in that motivated people from all over the world contributed to create the best possible code. It's an exciting way to develop algorithms, and a fun way to watch the process of innovation unfold.

REFERENCES

1. MATLAB On-line Programming Contest home page: http://www.mathworks.com/contest/.