# Algorithm Invention and Algorithmic Claim Evaluation with SCG

Greg Kerr
Northeastern University
Boston, MA
kerrg@ccs.neu.edu

## ABSTRACT

This paper will discuss the evaluation of algorithmic claims with SCG. There are two modes in which students in the class participated in the SCG: as a scholar, where they themselves made and evaluated claims, or as an avatar, where a program they wrote algorithmically made claims on their behalf.

This paper argues that SCG Scholar is the natural method by which students working in pairs solve their problems. By formalizing this method, however, with a proposer, opposer, and refutation strategy, it establishes the basis for SCG Avatar.

SCG Avatar benefits by pitting students against each other, particularly forcing top students to compete with other top students. The software itself could use some improvements that would make the tournaments a smoother experience. In particular, pitting students against each other with algorithms that have non-obvious optimal claims will allow greater competition.

## 1. SCG SCHOLAR

The manual SCG Scholar game is a natural game, one that students begin in Fundies when they first work in pairs to solve problems. Some students effectively discuss the issues at hand, proposing various solutions, eventually settling on the optimal choice. Many of these partnerships often run afoul, often caused by a lack of understanding of the implied refutation protocol, and/or defense strategy. I argue that for a freshman level course, the best solution is the one that correctly solves the issue and provides simple, readable code. These partnerships fail because the participants do not understand the defense strategy, as well a level of ignorance that hinders their ability to correctly analyze code for simple and effective design. Without sound protocols, the collaboration breaks down and no productive progress is made.

To respond to claims algorithmically a sound defense and refutation protocol is a necessity. The SCG scholar game allows students to begin considering the nature of an optimal defense strategy without worrying about writing code. The MMG tournament, in particular, shows that students can, with a very simple strategy, make claims about a problem and always defend them. They need not think when making claims: once the strategy is crafted it becomes a systematic process.

I do not believe SCG should be wholly implemented into the fundamentals course. For one, it may further intimidate students already struggling with programming for the first time. Secondly, the game forces students to face public scrutiny in regards to their performance. This is appropriate for an upperclassmen course, where students can be expected to set aside their egos, face the results, and do what is needed to improve performance. But nervous freshman need not face this added stress and scrutiny. Rather, it can be partially implemented into the course: not referred to as SCG or defense strategies, but students should be pushed to justify all claims they make to their homework partners. Through this it will become clear that certain strategies result in far fewer refutations.

## 2. SCG AVATAR

The SCG Court was indeed an ambitious experiment in class, and it was one that had its successes and failures. I will address first the deficiencies, then the successes, and then ideas for additional games in the future.

### 2.1 Deficiencies

The largest deficiency of the SCG Court was in the technical problems. In the first practice matches, the logs themselves were recorded incorrectly (the proposers and opposer were reversed) and the scoring was inadequate. It always bothered me that, even in the final tournament, optimal avatars may not have maximum points, because the opponents were

kicked out. I believe that if an avatar is kicked out, all of its future opponents should automatically be awarded 'wins' for the respective rounds to even out the scoring.

I want to briefly discuss the deficiencies in the valid function, as I believe they were essentially a non-issue in the tournaments. There were two potential pitfalls with valid: either a valid tree could be rejected in a tournament, or an invalid tree would be allowed. I never saw any instances of these problems in an actual tournament. In particular, to generate an invalid tree that would sneak through valid, it would have to have the correct depth and all nodes listed exactly once, but in the wrong order. It would actually be more difficult to generate one of these trees than to generate a valid tree. Nonetheless it was good to recognize the issue and make the needed improvements.

In general I advocate the use of JSON for communicating with the server. There are many arguments against JSON, such as that it is too heavy weight, but there are numerous advantages. Because there are standard libraries for JSON in almost every language, students are free to use any suitable programming language without having to write a parser. This allows them to fully implement their own local solution, including data structures, etc. This avoids the entire class of errors that was caused by students being unfamiliar with the pre-packaged Java code. Many students on the mailing list expressed frustrations caused by casting errors, parsing exceptions, etc. In addition, Java may not be the preferred language for many students.

There is one more reason JSON is interesting: if its easier for students to use a language of their choice, students can learn the advantages and disadvantages of their languages. For example, we can imagine a tournament where students have used C, Java, Python, and Perl. The C code may be the fastest but the team could share their experience with debugging to the class, and the Python group could take about the advantages of Python's multi-paradigms to Java, and so on.

## 2.2 Advantages

The biggest initial advantage to the SCG Court is that top students are pitted against each other in tournaments. This drives the top students to constantly perform higher to try and be the "best of the best." They also are able to learn from each other, by observing the other's performance in the games. In a typical homework assignment, there will be some benchmark of performance that the top students will meet. However in the SCG Court there is no such benchmark. The top students must then work harder to outdo the other top students.

Equally important, other students are able to learn from the top students. During tournaments they can observe the top avatar performance, and try and tweak theirs to match it. In addition, the code reviews reveal the secrets of the top avatars, showing the class a model for how to construct a winning solution.

## 2.3 Improvements

As stated previously, MMG would be best for the initial tournament. I agree with your rational for placing HSR first, but ultimately students seemed to struggle with the software package too much. In addition, a heavier emphasis should be placed upon deadlines the tournament. In the real world, especially at companies, being on time is very important. It's arguable that a true optimal avatar is not just one that defends all it claims, it is one that is ready by the deadline. These avatars should be declared the true overall winners.

During the HSR and MMG tournaments, one interesting fact was that avatars either worked or they didn't. There was very little in between room. To truly establish the "top students," contests should be chosen which do not scale well, or lack clearly defined optimal solutions. I now present a few examples.

### *2.3.1 24 Problem*

As discussed in your office, students could solve a variation of the 24 game. In the 24 puzzle[1], players are given a card with 4 numbers, and they must decide a series of steps to produce the number 24 from the given cards. A more generic version can be solved, where the quantity of numbers given, and the target value are are variable. There is, of course, a brute force algorithm to solve this problem. However, it scales very poorly as any brute force algorithm does.

This forces students to be more clever with their avatars. For one, they must try to make claims large enough to get their opponent kicked out, but that they themselves can solve. Furthermore, things such as memoization become important. This also opens the door for students to use distributed computations to speed up the calculations. Ultimately such a tournament should reveal the true best.

### *2.3.2 Sliding Puzzle*

Another good problem for students to solve is the slider puzzle[2], more specifically the Fifteen puzzle. In this game there is a square with 4x4 tiles in it, except one tile is missing. The tiles are then mixed out of order, and students must right a program to put the tiles back into the right order.

This game is fun and interesting not only because of solve, but because of propose. Students must try to create the most devious puzzles possible.

### *2.3.3 Playoff*

[1] http://en.wikipedia.org/wiki/24_Game
[2] http://en.wikipedia.org/wiki/Sliding_puzzle

Finally, I propose a playoff type system to deciding the winner. Consider the NFL. The top team is generally selected by the Super Bowl, because to make it that far, the teams must survive a brutal test of all the top teams in a playoff. Further more, the one on one nature of a playoff game adds further depth to the SCG Court. In the NFL it is common to analyze type of an opponent before a match, to modify your own strategy to counter them.

Consider a playoff of the Sliding Puzzle form. In this, top avatars will go through a series of one-on-one matches, with a day of rest in between. For the sake of their propose method, they must try to reverse engineer the opponents solve method by watching their performance in the games. By finding a pattern in the opponents weaknesses, they can tweak their propose method to suggest puzzles that will stump the opponent. Furthermore, they can analyze the opponents tendency to propose various puzzles. If they find a pattern, they can perform tune their own solve method to handle these cases as fast as possible.

I believe that by extending the SCG Court to include various games and a playoff as described, the competitive nature can be greatly increased.

## 3. CROWDSOURCING

SCG Court has an interesting potential as a crowdsourcing platform, where corporations or research groups can create playgrounds to see who can come up with the best solution to an algorithm.

The advantages to this are that its a low cost way for groups to have algorithms written. My biggest concern is that many corporations obsess over their intellectual property, and may be unwilling to crowdsource potential IP. Many large corporations also have many programmers on staff and can have some internally do the task.

Groups who may be more likely to take advantage of this, however, are small start-ups and research groups. These organizations are smaller, have tight budgets, and do not obsess as much over intellectual property. I would argue that a crowd sourcing solution would be much cheaper and lower risk than hiring a contractor.

To make this as useful as possible to potentially interested groups, I think again that JSON is a wise choice. It will decouple both the language and implementation: groups can implement a playground and avatar without actually knowing anything about the SCG implementation. As long as they see the spec of what they should send the server, and expect to receive back, the playground and avatars can be decoupled.