SUPLEX:

Style Underlying Predictions Lifting up Elo's Expectations

Andrew Hunter

May 10, 2008

Abstract

I present SUPLEX, a system for predicting the results of mixed martial arts matches. The system uses a number of techniques from graph theory and games, the most important of which relies on improving the well-known Elo system by separating fighters out by styles. I test the commonly held hypothesis that "styles make fights" and attempt to utilize it to improve my predictions.

1 Introduction

Most people agree that making money is usually a good thing. It is such a good thing, in fact, that gambling is a $85 billion industry in America alone [1]. Most of this gambling is on games of chance which, alas, we cannot help our fellow man profit from. On the other hand, people also gamble on other events, like sports, which are not entirely due to chance. It is quite possible, in theory, for me to use mathematical knowledge to predict the results of sports matches, and if I can do so better than Vegas–as they say, *beat the spread*–I stand to make a pretty penny. This paper summarizes SUPLEX, a system I built to do just that. I attempted to use my knowledge of graph theory, numerical optimization techniques, and standard approaches to games, along with a dose of domain knowledge, to produce accurate predictions for mixed martial arts matches, with varying degrees of success. In particular, I tried to examine a commonly held piece of wisdom in mixed martial arts, the idea that "styles make fights" and see if the data holds the same conclusion, with some surprising results. I will begin by giving some background on mixed martial arts and sport prediction.

1.1 Mixed Martial Arts

Mixed martial arts is a recently developed combat sport, whose first professional match occured in 1993. It has relatively few rules as compared to similar sports like boxing, only banning a few techniques–like eye gouging, punches to the back of the head, and headbutting–considered extremely dangerous to the health of the participants. For the most part, competitors can use their choice of fighting style, techniques, and tactics, and many martial arts find success in professional competitions, such as Muay Thai (Thai kickboxing), Brazillian jiu-jitsu, Sambo (a Russian grappling style), Greco-Roman wrestling[1], and many more. In fact, mixed martial arts resulted from the desire of experts in various martial arts to determine whose techniques were best in a straight fight.

It should come as no surprise, then, that styles are considered by MMA experts to play a key role in the outcome of matches. While the variety of styles employed by professional mixed martial artists makes the situation extremely complicated, in essence we deal with a situation similar to rock-paper-scissor: there is no best style, only relative advantages and disadvantages in particular matchups. One simple example might help make this clearer:

- *Kickboxers* rely on powerful strikes to knock out opponents, but often are inferior wrestlers and inexperienced on the ground.
- *Jiu-jitsu fighters* rely on taking an opponent to the ground, where they can apply dangerous submissions (joint-locks and chokes) but generally have sub-par striking.
- *Sprawl and brawl fighters* rely on their wrestling experience not to take down opponents, but to avoid being taken down themselves. They then use striking to defeat their opponents.

How do these three types interact?

- Jiu-jitsu fighters, while unable to stand with kickboxers, can usually take them to the mat, where the kickboxer will be unable to effectively strike. So, we find that jiu-jitsu fighters defeat kickboxers by submission.
- Sprawl-and-brawlers, however, have strong wrestling pedigrees, and can defeat a jiu-jitsu fighter's takedowns. Since the fight then remains standing, the sprawl-and-brawler's superior punches and kicks usually defeat the jiu-jitsu fighters.
- Kickboxers, on the other hand, have better striking than sprawl-and-brawlers. Since the kickboxer never attempts takedowns, the "sprawl" never comes into play, and the kickboxer should usually knock out the sprawl-and-brawler.

[1] SUPLEX is in fact named after a Greco-Roman technique.

Thus, most experts do not believe we can simply assign ratings or strengths to various fighters; we can't say that Chuck Liddell is just a better fighter than Tito Ortiz because Chuck won, because Liddell's sprawl-and-brawl is designed to defeat submission fighters like Ortiz.

1.2 Prediction Background

This is a pity, because one standard technique in predicting the results of sports in games is the Elo system, which relies on assigning each fighter a numerical ranking, and assuming that a higher-ranked fighter can usually defeat a lower-ranked fighter. Invented by a Hungarian mathematician to help rate chess players, the system has found phenomenal success both in chess and elsewhere, in various computer games, Go, Scrabble, football, basketball, and soccer[2] just to name a few. The only requirement for Elo to work with a particular game is that it is possible to assess the overall *strength* or *rating* of a player. Suppose player A has a rating of R_A points, and B is rated at R_B. Then, the Elo model, derived from the logistic distribution, tells us that

$$P(\text{A wins}) = E(A, B) = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}$$

Thus if $R_A = 2000$ and $R_B = 1600$, we find that A has a $\frac{10}{11}$ chance of winning a match between them. While simple, this model often accurately predicts match results, explaining its success.

However, how do we determine A and B's ratings? Typically, new players start out with an arbitrarily assigned rating, traditionally 1600. Then, as the new player plays games, winning and losing some, we compared his expected performance with his actual performance, and adjust his rating commensurately. For example, if a 1600-rated players plays and beats a 2000-rated player, he has upset the favorite. Since he was expected to win $\frac{1}{11}$ of his games, and actually won 100% of them, he will gain $k\left(1 - \frac{1}{11}\right)$, where k is a parameter, often 16 or 32, that determines the flexibility of ratings. Similarly, his opponent has underperformed expectations, and will lose the same number of points.

This method allows ratings to gradually change as players continue to play and improve or get worse, but in practice we can use other methods. If we have a large set of players and match results, we can simply iterate over the matches many times, adjusting with a very small k, and the ratings for these players will rapidly converge to the "best" ratings for explaining the results.

It should be clear why we need an absolute measure of strength for Elo to work. If we have three groups who beat each other transitively, each player will win half his games and lose half, and most ratings will stay around the 1600 level, without any indication that the jiu-jitsu fighters, for example, are much more likely than 50% to beat the kickboxers. So, if "styles make fights" is indeed a correct observation, we should *not* find Elo an effective method of making predictions.

1.3 Goals

If Elo isn't a good idea, what should I use? Professional bookmakers use domain knowledge, intuition, and many other factors to set odds on MMA fights. Predictions are made based off estimates of strength, athleticism, psychological preparation and previous results, among other things. I, however, want to make my predictions by computer, and it's hard to explain to a computer that, say, my intuition tells me that this fighter beats that one for some subtle reason. There is a lot of data, however, in *past performance*: despite only existing for 15 years, professional mixed martial arts has had many matches, and records of all of them exist. Moreover, it's easy to tell a computer event results. It should be relatively simple to convince a computer to make predictions just from examining the more than 45,000 professional fights on record, and this seems an admirable goal. Can we make accurate predictions using just that data?

I do not expect that this method should be able to beat the spread. If the only thing I consider, when asking who will win a future match, is the result (win or loss) of various past fights, how can I conceivably compete with a bookmaker who consults that information, and many other useful facts about the fighters in question? I even know less about the results of fights than this hypothetical bookmaker; for simplicity, I only

[2]In fact, Elo is the *official* rating system for at the very least chess and women's soccer.

consider wins and losses, and there is generally considered to be a great difference between, for example, wins by submission or knockout, and wins on time (by a judge's decision.) So, I do not believe it is reasonable to expect my methods to be profitable directly, only interesting. Hopefully, my predictions should be at least better than random.

In particular, I am interested in considering applications of graph theory to this problem. I have data consisting of a large set of fighters, and every fight that has taken place between members of this set. There is a natural embedding of this as a graph; we have a vertex for each fighter, and an edge for each fight, from the winner to the loser. There is an extensive literature on interesting ways to analyze graphs like this; shouldn't I be able to exploit well-known graph algorithms to produce good predictions? In the next section, I will describe how I went about this problem.

2 Methods

The first challenge I faced pruning the provided data. The set of fighters and fights I began with had 45,336 fighters in it, and 65,330 fights. This presented two problems:

- It was too big to practically work on. As we shall see, many of the algorithms I need to use are computationally expensive and completely intractable on a graph of this size.
- It contained a lot of noise. The vast majority of these fighters had fought once, and in small events no one had ever heard of or cared about. Not only was I not interested (since no one would bet on it) in predicting the results of West Virginia Hilltown Fight Night, the noise this introduced obscured real trends–in short, I cared a lot if someone had ten wins against high-level expert fighters, but not if someone had ten wins against nobodies never seen before or since.

I used simple graph techniques to reduce the graph to one I "cared about." I picked a small integer k, then found the set of vertices (fighters) with k or more edges (fights.) I took the induced subgraph of that set (threw away all the other fighters, and all the fights that involved those unimportant fighters.) I then repeated this process til I got to a fixed point, a set of fighters such that each one had fought other people in the set at least k times.

This worked extremely well. With $k = 5$, I reduced the graph to 2,196 fighters, and with $k = 8$, I got 332 fighters. These sizes were much more computationally tractable (if barely, in the 5-graph case.) Moreover, I found that as I hoped, for the most part I threw away unimportant fighters and kept important ones. The top fighters, which I knew from outside experience to be extremely good and also wagered on, generally stayed in the smaller graphs, whereas the pruned fighters were the ones I had never heard of.

Next, I had to come up with a model for doing predictions. I wanted to test the *style hypothesis*, or if styles made fights, so I made an observation:

- Elo rankings won't work, in theory, because fighters of different styles can't be directly compared. If we calculate Elo rankings inside a specific style, it should work well.

Thus, if I break fighters down by styles, I should be able to find Elo rankings within styles, and easily predict whether one kickboxer will defeat another. What about predictions for fighters in different styles?

Well, this model was predicated on some styles "beating" others. If that's accurate, I should be able to find that on average, style A beats style B. I can then combine the proportion of A versus B wins with the A-fighter's ranking within his style and the B-figher's ranking within his style to find an overall prediction.

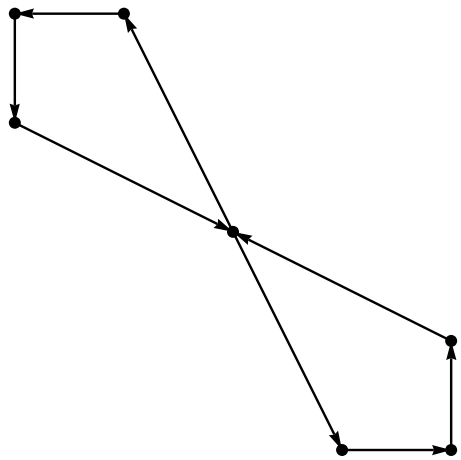
How could I find styles? Well, the first obvious method would be to examine what fighters claim their styles to be, but that would work poorly. Styles are poorly reported; many fighters simply don't list one, and many others give either very general styles, like "mixed martial artist", or very specific and more marketing than reality based styles, like "thug-jitsu." The key problem in this project would be *inferring styles*, and I made another observation:

- The problem lies in non-transitivity, with kickboxers beating sprawlers beating jiu-jitsu fighters beating kickboxers. These non-transitive results translate to *cycles* in the directed graph of who beat who; in

a set of fighters of the same style, we should find very few cycles. Thus, I should try to *partition the fighter graph to minimize cycles within the induced subgraphs.* Each set in the partition will then be a "style."

This, however, is harder than it sounds. To evaluate the goodness of a partition, I need to know how many cycles each induced subgraph contains. However, the best known algorithms for counting or enumerating cycles [2] are at *least* exponential in the number of vertices. Moreover, we have good reason to believe[3] that the problem of partitioning is even harder–completely intractable. Thus, we must resort to heuristics and approximate solutions.

To find a heuristic, I note an interesting property of adjacency matrices. We see immediately that A_ij,



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 1: A small directed graph A

the entry in the ith row and jth column, is the number of paths (here, either one or zero) of length 1 from vertex i to vertex j, since there's a path of length 1 if the vertices are adjacent and not if they are not.

A little experimentation will reveal a more useful fact, which isn't difficult to prove: A^n_{ij}, the entry in the ith row and jth column of the nth power of the adjacency matrix, is the number of distinct paths of length n from i to j! (We can prove this trivially by induction, examining the entries multiplied in a single matrix multiplication.) So, if there is an n-cycle in the graph that includes vertex i, we will find $A^n_{ii} \neq 0$, and if no such cycle exists, it will be zero.

But how do we exploit this to count cycles? We *cannot*, for example, say that if $A^8_{33} = 4$ that there are 4 8-cycles involving vertex 3. We will see a non-zero value there from any two 4-cycles, or any 4 2-cycles, or any 2-cycle and 6-cycle, or... Moreover, there's something of a combinatorical explosion in paths, unsurprisingly, making the values of A^k_{ii} get relatively large as k grows so long as there are *any* cycles involving vertex i.

Nonetheless, let us try to find a useful heuristic. If there are few cycles, we should find a low value along the diagonal of the powers of the matrix:

$$\mathrm{tr}(A) + \mathrm{tr}(A^2) + \mathrm{tr}(A^3) + \ldots$$

But since trace is a linear operator, that's just:

$$\mathrm{tr}\left(A + A^2 + A^3 + \ldots\right)$$

[3]Personal communication from Professor Chen at Pomona.

We should probably weight down higher powers, since we care less about longer cycles (they say less about non-transitivity of rankings, and we'll find way more of them due to the combinatorical explosion.)

$$\operatorname{tr}\left(A + \frac{A}{2!} + \frac{A}{3!} + \dots\right)$$

And since we're finding a partition to minimize this quantity, it's OK to add something constant to it:

$$\operatorname{tr}\left(I + A + \frac{A}{2!} + \frac{A}{3!} + \dots\right)$$

But that's just the matrix exponential!

$$\operatorname{tr}\left(e^A\right)$$

Thus, I hoped that if I found a partition so that I *minimize the sum of the traces of the matrix exponentials of the adjacency matrices of the induced subgraphs*, I will find a partition with few in-set cycles, and thus be close to finding styles. I tried to find these partitions for 3, 4, and 5 styles (thus for 3, 4, or 5 sets in the partition.)

Finding partitions, however, still wasn't easy. As a reference, note that the number of partitions of a set of n elements into k non-empty set are described by the *Stirling numbers of the second kind*, S_k^n [3]. For the smallest graph I consider, if I break it into (say) 3 styles, we have:

$$S_3^{332} \approx 10^{157}$$

This is staggering. It's as if I was trying to find a maximum of a function on a unit cube in $\mathbb{R}^3$ to a resolution of 10^{-52} or better in each direction. Throw in the fact that the function is highly nonlinear, barely continuous, and most importantly, that evaluating each point takes *more than a second on a high-powered server* in Mathematica, we see the difficulty.

Nonetheless, there are well-known methods for finding optima of functions, and I assumed (correctly, as it turned out) that standard methods *could* be made to work. I decide to use the simple method known as *random-restart hill climbing*: pick a point, move along the gradient to a local optimum, and then start again somewhere else randomly. Repeat until you have enough local optima you like, and then pick the one you like the most.

The problem with hill-climbing was simple: the adjacency factor, so to speak, was far too high. Even in $\mathbb{R}^3$, there are only (so to speak) 6 directions I can go. Even with the simplest possible model of adjacency I could think of, calling two points adjacent if I could convert one into another by moving one vertex between sets:

$$\Big\{\{1,2,3\},\{4,5,6\}\Big\} \sim \Big\{\{1,2,3,4\},\{5,6\}\Big\}$$

I still would have more than 300 points adjacent to each point, and with larger graphs, proportionally worse. Given that it took noticable time to evaluate each point considered, this meant that random-restart hill-climbing never got to the restarts: I couldn't even make it all the way up a hill, even given hours to calculate.

However, I noticed an interesting pattern in the motion. The random points would quickly see great improvement, then slow down, taking longer to find any move that would improve the heuristic, and finding smaller improvements when they did exist. I interpreted this as my algorithm quickly getting *near* minima, but having trouble finding exactly where they were.

Oh well; near minima is still pretty good, especially considering I'm already using a rough heuristic . Remember that I am dealing with a highly intractable problem–I was ecstatic that I could even *try* and get approximate solutions[4] and thus I was OK with this behavior. Since I didn't want my algorithm to hang up

[4]This is especially true considering I have reason to believe the counting problem is #P-Complete, a complexity class that is not only hard to solve, but considered to consist of "hard to approximate" problems. In the future, I plan to avoid picking class projects that summarize to: approximate a problem that can't be approximated. Just FYI.

trying to find exact minima, I decided to change my method somewhat, implementing an small optimization I have termed ADD-SEARCH, since I have not yet found it referenced in optimization literature.

The name comes from the behaivor, which models a child with attention deficit disorder. Just like I would in the situation, I tell the computer that if it can't quickly find an improvement nearby the current point, get bored and assume it's probably a minimum. So, instead of enumerating *all* neighbors of a candidate point, I pick N (a configurable parameter) neighbors of random. If any improve, I move there and start again, just as in hill-climbing. If not, I terminate the search, reporting that point as a local minimum, and go to the next random starting point. I also added a parameter δ, defining the minimum improvement "worth investigating". By tweaking N and the number of starting points considered, I can change how the search functions. With higher N and lower number of starting points, the search is "deep": I carefully investigate a few starting points. The other way around, the search is "wide": I find very rough minima near a *lot* of starting points. In practice, both ways work; with any reasonable values for parameters, I get about the same "optimization per hour of computation", so to speak.

This worked extremely well. Though it took a lot of computation, I was able to generate a lot of very promising candidate partitions with very low values for the trace of the matrix exponential. At that point, my job was simple. I took each subset in the candidate partitions, applied iterated Elo to find rankings in that "style", and decided how to combine relative Elo for two sets and the overall performance for those two sets.

This I solved with a simple linear regression. For each pair of styles/sets A and B, I singled out the fights in my database with one fighter from A and one from B. For each fight, I created a pair (p_e, r), where p_e was the Elo prediction (a probability A would win) and r was a result (1 if he did, 0 if B won.) I then used a linear model, stating my overall prediction would be:

$$p = c_0 + c_1 p_e$$

And optimized for the ideal parameters c_0 and c_1. That was enough information to do predictions. If asked to predict the result of a fight between a and b, I would:

1. Look up the styles A and B for a and b.
2. Use the Elo formula to make a prediction e_p for a fight between a and b.
3. If $A = B$, just return that prediction.
4. Otherwise, look up the corresponding values c_0 and c_1 for that style pair, and return $c_0 + c_1 e_p$.

Note that this method allowed me to integrate a simple Elo-only model easily. If the only factor to consider was Elo, with no styles, I just took the list of fighters and considered them all members of the same style. This simplified the coding greatly.

3 Data

At this point, I had a number of models (determined from partitions into different number of subsets, of differently sized source graphs, and often several candidate partitions in each case[5].) It was time to determine which models accurately predicted fights, and to what extent.

I tested my various models on a small but representative set of fights with known results. While I could assess success on various criteria (statistical goodness-of-fit tests, for example) I decided to exploit my purpose here–gambling predictions–to evaluate predictions. I took the Vegas odds on these fights, and compared them to the odds my models produced. When my model thought a fighter was more likely to win than Vegas did, it placed a "bet" on that fighter. I then added up the wins and losses each model incurred per dollar bet, and considered that the model's performance.

[5]For simplicity, when I have multiple candidate partitions that correspond to the same model–say, two possible breakdowns of the smallest candidate graph into three sets–I present the best data from any of them.

I was immediately able to discard some models. For example, the partitions of te larger (3000-fighter) test graph) produced uniformly awful predictions. The simplest explanation here, and the one I favor, is that I simply hadn't found sufficiently good minima. This was something I feared likely; on the larger graph, evaluating a single point often took as long as 10 seconds. I was able to do only very minimal optimization, even with ADD-SEARCH. Despite finding significant improvements from the initial values of the heuristic, I think I just didn't find good *enough* results. I'll name the remaining models for convenience:

Model name	Description
ELO-ALL	Elo run on the *entire* graph, all 45,000 fighters
ELO-LARGE	Elo run on the large test graph (with 3000 fighters)
ELO-SMALL	Elo run on the small test graph (with 300 fighters)
3-PART	The small graph partitioned into 3 styles
4-PART	The small graph partitioned into 4 styles
5-PART	The small graph partitioned into 5 styles

The rest of the results were both interesting and inconclusive, and in a particularly surprsing fashion, depended heavily on the betting model used–most discussion of betting focuses on making accurate predictions, and assumes the correct bets follow naturally. If every time we identify an edge (a bet where Vegas has set the odds such that we think there's positive expected value in taking it; in other words, being offered 2:1 odds when we think the fight is 50/50) we bet some constant amount, we see results like this:

Model	ELO-ALL	ELO-LARGE	ELO-SMALL	3-PART	4-PART	5-PART
Winnings per dollar bet	0.111	0.111	0.162	-0.094	-0.094	-0.292

However, What if we bet proportionally to the edge that we found? In other words, if we are offered 1:1 odds, why not bet more if we think one fighter has a 90% chance of winning as opposed to only a 60% chance? Surely both are good bets, but we're more sure of the edge offered to the fighter if we think he'll win 90% of the time. Betting like that gives us:

Model	ELO-ALL	ELO-LARGE	ELO-SMALL	3-PART	4-PART	5-PART
Winnings per dollar bet	0.104	0.127	0.179	0.097	0.229	-0.166

We see that this betting method greatly benefits the style models, without a similar benefit to the simple Elo models. This is fascinating, but I have yet to determine why it happens. There are other conclusions I can come to, however, from these and other data I can collect:

- My hypothesis that throwing away "unimportant" fighters is valid. I get better predictions from the *smaller* graphs. I can't take this to extremes–in the 300 fighter graph, I've already started to throw away fighters whose results I care about predicting–but in general, it is safe to prune the graph.
- 4-partitioning the fighter set seems to work best (the data show this somewhat, since the best partition result comes from a 4-partition, but other data I have make this even clearer. For example, these are the results from the best partition models–however, most of the 4-partitions are better than most of the 3-partitions consistently, even if the best 3-partition sometimes is as good as the best 4-partition. I thus conclude that *if* my partitions accurately represent styles, that the underlying data support there being exactly 4 styles.

So, SUPLEX is able to predict MMA to some extent. My limited testing and small positive EV prevent me from conclusively claiming that the system works, but I think I can say that SUPLEX isn't totally broken and can produce at least some good predictions.

4 Conclusions

In short, I exceeded my expectations: I not only did better than random chance, I came close to making money with several models! Nonetheless, I was disappointed somewhat, because it was clear that as implemented,

the style hypothesis wasn't a big win. Depending on the measurement, it might be improving predictions or might not, but it wasn't a clear positive.

There are several reasons why this could be. The simplest would be that it's wrong, that we can simply ignore styles and assign overal ratings, but I refuse to accept this. It's more than conventional wisdom; we can see it in practice in every fight we watch. I in fact have *seen* mediocre sprawl-and-brawlers decisively defeat some of the best jiu-jitsu practictioners in the world, simply due to their gameplan having an advantage against jiu-jitsu. Styles *surely* change results; I just need to figure out how. The two more reasonable possibilities I can think of are also quite plausible: either my heuristic/optimization isn't good, and I'm not finding proper styles, or I'm using style information wrong. The second is particularly interesting to consider. Remember that as is, a fighter's rating is determined solely by his fights with other people using his style. This penalizes fighters who, due to the vagaries of matchmaking, mostly fought other styles–some fighters, in fact, have *no* in style fights in several of my models, and thus have *no* information determining their Elo rating. Thus, I think it's possible I'm throwing away more information in style breakdowns then using them in my model gains me. Having the style information change predictions in some more subtle way might be more useful.

5 Future Work

While I have yet to make predictions I would personally put money on[6], I am very pleased with these results not because I trust them, but because they show these techniques have promise and point out several places I could go to improve them. Some possibilities for future work include:

- *Improving the betting model.* Whether to trust overal Elo or style splits, as we saw, depends drastically on how we bet–constant, or in proportion to how big our edge is. What's the *right* way to bet? Is there a right way, in terms of game theory? There is fascinating work to be done here.
- *Drastically improve testing.* My conclusions are predicated on a small set of fights that are partially contained within the training data. This is justified because if the model can't predict the values it's trained on, it can't reasonably predict anything, and more justified by the sparse available data. Georges St. Pierre, the consensus best mixed martial artist alive, has all of 18 fights under his belt, and in my smaller data set, he has only 11. Compare this to chess players, whose Elo rankings are determined from hundreds or thousands of matches–I really can't afford to throw anything more away to reserve as testing data. Nevertheless, more testing with more data (for example, as new fights happen, which won't be contained in my database) might allow me to make conclusions with more conviction.
- *Determine styles differently.* While I did very interesting work, with some cool results, using the matrix-exponential heuristic, I'm not convinced that it produces the best results here, and most importantly, it's hard to see *why* it picks the partitions it does. Fundamentally, what I'm looking for are two sets of fighters, A and B, where fighters from A do better (or worse) than otherwise expected against fighters from B. There are many ways I can conceive to find such sets, and I think some of them may have more promise.
- *Utilize the style hypothesis differently.* My conclusion above was that I paid too great a price in ranking fighters by segregating out all fights that were out-of-style. Perhaps I can more subtly and effectively use the style information I gain, as opposed to crippling my overall rankings.
- *Optimization.* It is said there are two rules of optimization: don't do it, and don't do it *yet*. I think I'm at the "yet" stage here. As mentioned, most of my partition results from the large graph were rather

[6]In case you're more impulsive than I am, B.J. Penn is facing Sean Sherk in two weeks and is favored 2:1, whereas my models consistently give Sherk a 3:2 edge. On the same card, Lyoto Machida is also a 2:1 favorite to beat Tito Ortiz; my models consistently call him a 10:1 favorite or better. Just saying, that's all. If you're dumb enough to actually bet on my advice, you deserve what's coming to you, though...

poor, and I conjectured that this was due to not having very good optima there. That was due to the high cost of optimizing on such a large graph–I could search much better on the smaller graph since it was cheaper to evaluate. If I rewrote my search algorithms to avoid using Mathematica (whose graph libraries are, in my opinion as a programmer, *awful*, being slow, buggy, and awkward) and instead use a fast C/C++ graph library, hopefully one that has been well parallelized, I would be able to do much better search. This would pose a significant opportunity cost, as I make heavy use of Mathematica's numerical computation routines, which are pretty good, but might be worth it.

Thus, while I don't yet have a million dollar betting application yet, I think the approaches SUPLEX uses are promising, and may yet create such an application eventually.

6 Acknowledgments

I would like to thank Professor Yong for advice and support throughout the project, Professor Tzu-yi Chen for help with graph algorithms and complexity estimates, Steven Sloss for Mathematica advice, and Phil Miller for help with shell scripting and nice-looking LaTeX, and Professor Groves for putting on Shakespeare and eating my schedule for the entire second half of this semester.

References

[1] American Gaming Association. Gaming revenue: Current-year data. http://www.americangaming.org/Industry/factsheets/statistics_detail.cfv?id=7.

[2] Donald B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Comput.*, 4:77–84, 1975.

[3] Wikipedia. Stirling numbers of the second kind. http://en.wikipedia.org/wiki/Stirling_numbers_of_the_second_kind.