



A Model for Analyzing Contests

Introduction

What follows is a description of a mathematical model for analyzing contests of all kinds, using only the array of points scored by the participants against each other. The simplest application is the ordinary athletic league, but the model will also supply the mathematical basis for solving other types of contests, including the long-standing *voting* problem in elections where each voter can vote for more than one candidate. On the way to the voting problem, it also solves two other problems which turn out to be closely related to voting. These have been given the shorthand names *roommates* and *marriage*, referring to making pair matches of participants by means of their expressed preferences. Other problems related to voting and choice are solvable with contest analysis and even game theory--seemingly unrelated to choice--yields nicely when viewed in a certain way. The contestants need not be at all similar; first application of this method was to military forces in combat, where the participants are vehicles or weapons of many types engaging each other. (I developed this model for the US Army, and the Army has been using it in combat simulations since 1985.) Even the performance of legislators, as evidenced by their records of voting on bills, is amenable to very detailed analysis. The link to "topics" at the end of this report will take the reader to many examples of various types of problems.

Contests in which the participants compete in a single dimension--as with a measure of time, distance, or weight--are excluded from this analysis method, since comparison of participants is immediate. On the other hand, judged contests are included, with the judges becoming part of the process.

Origin of the Model

This model began in the 1980's with the discovery of a set of equations for determining the strengths of weapon systems in computer simulations of military combat. It was thought that the record of losses of vehicles and weapon systems by cause (the so-called *scoreboard* or *attrition matrix* of an engagement) would contain sufficient information to compute the relative strengths of the engaged participants. The only criteria to be satisfied by the set of equations were *monotonicity* and *groupability*. The first of these means that any increase in losses caused by a participant will produce an increase in the strength of that participant. The second criterion means that identical participants can be put into any number of arbitrary-sized groups without changing the results of the computation. A unique equation was able to satisfy these two simple criteria and the set of equations of that form was accepted by the US

Army as an algorithm for inclusion into major combat simulations. The algorithm has been run millions of times over many years. I have been aware for a long time that many types of contests, not only military engagements, should use it. What probably kept it from being picked up by workers in fields outside of combat analysis was the absence of a derivation or good explanation for the form of the basic equation. Starting with the equation and working backward to an explanation or physical analog has been difficult, but now a good derivation can be presented and it is time for the algorithm to be brought into general use. It is unique, significant, and quite unanticipated by those who could most benefit from it.

General Features of the Model

Surprising as it seems, the standard way of determining winners and losers in contests by comparing their point scores is capable of giving misleading results. Points can take many forms but, no matter what form they have, they must be weighted in a certain way before the contest can be properly analyzed. The weighting factor which is to be applied to each point in a contest is the *degree of difficulty* associated with producing that point. Observers of contests often comment about points being easy or hard, and it is intuitive that a point is easy if it is made by a strong competitor against a weak one. This introduces the concept of *strength* in a contestant, which this model quantifies by solving a system of equations called the contest algorithm.

Since the strengths of the contestants are the unknowns of a system of equations, they are designated by X variables, where X_i is the strength of contestant i . The matrix of the points in the contest is called A_{ij} , the number of points scored *by* contestant i (the attacker) *against* contestant j (the defender). A point becomes more difficult to score when the defender becomes stronger or the attacker becomes weaker. The algorithm uses another quantity, designated by f_{ij} , which is the fractional defense involvement of contestant j with contestant i --the fraction of all points scored against a particular defender which are attributed to a particular attacker. To explain the form of the algorithm, it is necessary to go through the derivation of the equation for X_i which is based upon a mechanical analog. This mechanical model for a contest is described elsewhere on this Web site, and those with an engineering bent are encouraged to find it there ([Derivation](#)) along with an accompanying derivation of the contest equation. The next paragraph is just a brief sketch of the arguments leading to the equation.

The strength of a contestant is the magnitude of a force which can be given a direction as a *vector* in a space of n dimensions, where n is the number of participants in the contest. Every participant has a direction in this space (an axis), and directing a vector at a participant means pointing the vector in his direction (parallel to his axis). Every participant also has two vectors, equal in length but generally different in direction. The length represents the strength of the participant and the two directions represent the directions for attack and defense. In attack, a contestant applies his force in the directions of those he is attacking, and these directional applications become the orthogonal components of a single attack vector. For defense, the contestant applies his force in the directions of those attacking him, producing the components of his defense vector. Whenever contestant i produces an attack-force

component in direction j, contestant j will produce a defense-force component in direction i--the two directions being orthogonal. This orthogonality of force directions is characteristic of braking, where a braking force is used to press against a friction surface to stop a moving object. *Contest scores turn out to be analogous to friction constants in brakes.*

The contest equations were *discovered* years before they were *derived*. The imaginary machine referred to above was thought of fairly quickly but the proper derivation using it came into the picture much later, giving the present form of the equations in the algorithm. Each participant has a separate equation for his strength, as shown here for contestant i. The meanings of the symbols were given above, and \sum_j or \sum_i are summations over all j or all i.

$$X_i = [\sum_j A_{ij} \cdot f_{ij} \cdot X_j]^{1/2} \text{ where } f_{ij} = A_{ij} / \sum_i A_{ij}$$

These equations are easy to solve numerically, even for very large numbers of participants. The vector *components* are found after solving for the X values. The square root of each term in the j-sum is the component of attack vector i in direction j. It is seen that the strength of an attacker depends upon both the score against a defender and the strength of that defender. A derivation of the equation following from physical principles is the best source of confidence for the method. The mechanical-analog derivation on this site was referred to above.

The attack vectors are used to compare contestants, but only after taking account of the vectors' differing orientations. By forming a vector sum of all of the contestants' attack vectors, we obtain the *contest vector*. Then the projections of the individual vectors upon the contest vector can be used in a scalar way to compare contestants. A suitable name for this vector projection might be *effective strength* or *applied strength*. An analogy would be a group of men applying various tensions to ropes attached at various angles to a rock being pulled out of the ground. The efforts of the men can only be compared for the *task at hand* by looking at the resultant vector and the contributions of the men to it (i.e., the projections of the rope-tension vectors upon their vector sum).

Going beyond the relative effective strengths, other information about the contest can be obtained from this model. The cosines of the angles between attack vectors will show how the contestants support each other, individually or in groups. Contestants which are identical in the attack will have their attack vectors all parallel (cosine equal to 1 for any pair), showing that they fully support each other in attack against their targets. Contestants on opposite sides of a two-sided contest (as in combat) will have 90-degree angles between their attack vectors (cosine equal to zero), showing that they give no support to each other. Also, in two-sided contests, the combined attack vector of one side (vector sum) will be orthogonal to that of the other side.

Groupability and the Form of Scoreboard Entries

Groupability is the requirement that participants with identical behavior in both attack and

defense be capable of placement into a single group or any number of various-sized groups with no change in results. This feature was needed for combat analysis, where the scoreboard entries are fractional losses of type- j participants due to the action of type- i participants. If the output of the computation is the strength of a tank, for example, it is unacceptable for the strength to depend upon whether the engagement has one group of 10 identical tanks or, say, a group of 4 and a group of 6. If the tank in the group of 4 were to differ in some slight physical way from the tank in the group of 6, we expect the strengths in the two groups to differ slightly, with the difference disappearing when the physical difference is removed. (This can happen simply by removing something from the opposing force to which one tank type can fire while the other type cannot.) For groupability to be present, the equation for computing X must belong to a particular family of functions and also the scoreboard entry must be expressed as *points per member of the defending group*, just as the scores in the combat situation are fractional losses or losses per group member. This is of no concern in the majority of applications, where no grouping of participants is done and all groups have one participant. Participants must be groupable if their scoreboard *row* entries are all in a fixed proportion to each other and *column* entries are all equal to each other (where the rows pertain to the participants as attackers and the columns as defenders). The new scoreboard for grouped participants is then formed by combining the proportional rows (into one row for the group) and deleting all but one of the identical columns (leaving one column for the group).

Application to Pair Matching

Some interactive assemblages require some thought in forming them into contests and in specifying what are to constitute 'points'. An illustration of this is the *roommates* problem, in which individuals are to give preferences for other individuals as potential roommates (or pairings of other sorts). From these preferences, some mechanism is then supposed to select a 'best' set of pairings. (This problem has never really been solved.) This problem does not look like a contest, at first, because people are not trying to defeat each other, but it is a 'reverse' contest in the sense of mutual support--individuals giving support to pairs and pairs giving support to individuals. There is a team of individuals and a team of potential pairs, and the two teams are engaged with each other through their preferences--individuals for pairings and pairings for individuals. To arrive at a scoreboard for this problem, we must first have the participants give their preferences in the form of *preference distribution functions*, i.e., a single vote split among the potential pairings in proportion to preference. The scoreboard for the problem will consist of two sections, individual vs. pairing and pairing vs. individual, and the entries in the scoreboard will be the preferences. The preference of a *pairing* for an *individual* will be the same as the preference of the potential *other member* of the pair for the individual.

The contest analysis algorithm will work from the scoreboard to find the strength vectors of all participants--the individuals and the potential pairs--but only the pair vectors will be used to select the roommates. The objective of the process is to put compatible individuals together, and the vectors of potential pairs have the information to do this. The logic goes this way: (1) The vector of a pair is the sum of two orthogonal components, each component pointing in the direction of one of the potential roommates. (2) Each of the two components

will have a projection upon the contest vector. The best pairs will be those whose two vector projections are large and nearly equal. (3) The way to quantify the quality of a potential match is to use the product of the two candidates' projections.

It turns out that the product of the projections is proportional to the product of the weighted preferences of the candidates for each other, where the weights are the magnitudes of their computed strength vectors. The roommate pairs are picked in sequence, based on the quality of their potential pairing.

A variant of the roommates problem is the *marriage* problem, in which two populations (as in men and women) are trying to be matched into pairs according to preference. This is no different from the roommates problem, since either type of problem will have zeros in the scoreboard for unpreferred pairings. The same solution procedure is followed as before.

Extending the method to the problem of higher-order matchings (triples, quadruples, etc.) is straightforward, since all it takes is a way for a potential grouping of any order to express its preference for an individual. There is a way to do this, and it is discussed in the topic called [Examples](#) .

A discussion of something called *stability* is needed now, because it figures very strongly into the way the marriage and roommates problems have been addressed in the past and because it will turn out to be not applicable to the proposed new method of solution.

Stability in Pair Matching

Stability refers to a matching solution in which pairs are created with such preferences that no two pairs can trade partners and have at least one member of each new pair obtain a better partner than before. The assumption behind past methods of solving matching problems is that the solution must be stable, and this assumption can make use of only a *rank-order* specification for the preferences (i.e., without regard to *how much* better a new partner would be). For the marriage problem, the method has one side *proposing* a match and the other side *accepting* in a bargaining sequence. The proposer starts high in preference and must then go to lower preferences as various acceptors reject him. The acceptor, meanwhile, is holding out for better and better proposals. The end of the process is a stable set of matched pairs, *but a different set is obtained when the proposing and accepting sides are reversed (using the same input data)*. Besides these two, many more stable solutions are typically present in a problem of any size, but finding them is more difficult than finding the two bargaining solutions. For the roommates problem, one does not have two distinct sides to form the bargaining sequence, but the method again searches for a stable solution (of which there are usually many, but sometimes none).

If strengths of preferences are admitted into the analysis, the following argument can be used to show why a proper solution need not be a stable solution: Suppose we had a marriage problem with two men and two women, and only one of the participants had a preference, the other three being indifferent. The solution would then be the matching of the one with the

preference to his or her preferred partner, with the two left over going with each other. Next, allow the three indifferent ones to have slight preferences, each favoring the potential partner not previously assigned. Now there would be three matched with their second choices and one matched with a strong first choice--an unstable solution, but fully expected. The three have given up their weak preferences in deference to the strong preference of the fourth. Now, if the three weak preferences are allowed to become stronger in small steps, a point must be reached where the matches flip over to their stable configuration: three paired with their first choices and one paired with a second choice. To find the point at which the flip occurs, an algorithm is required, and the contest model has it. The fact that the contest-model solution contains some unstable pairs is not a disqualifier for the contest model; it is only a reflection of reality when preference strengths are included in the problem. While the contest model solves every problem of the marriage or roommates type, it may leave some participants (mainly at the low-popularity end of the pairing sequence) who could improve their *ordinal* preferences in switches with others. However, their preference *strengths* may be improved less than the ordinal improvement would suggest. Switches of unstable pairs will be to the detriment of the ensemble, because the solution of the problem is optimum. So where does that leave us with regard to the free-agent aspect--the idea that any participant should be free to make his best bargain at any time? I think that negotiation should be over preference strengths before the problem is run. Then, when the players agree to take part in the process, they should also agree to be bound by the results.

If 100% stability is lost in use of the contest model for solving matching problems, some rather nice gains are achieved. The most important of these is the uniqueness of the solution, indicating optimization. Stable solutions are far from unique (sometimes running into the hundreds) and only the two bargaining solutions for the marriage problem are normally considered for use. They are the easiest to find and, even with them, it is not clear which is preferable. But why should it be either one of them, or any particular one of the many possible solutions? The roommates problem does not even have two bargaining alternatives to start with, as candidates for the 'true' solution. Another gain is the ability to use a far better expression for preference than simple order. Yet another gain is the ability to solve all matching problems, avoiding the need to give some an 'insoluble' label. Higher order matching (triples and up), readily done with the contest model, cannot even be addressed with bargaining or any other method.

If all one has for data are ordinal preferences, one might think that conversion to cardinal preferences has an element of arbitrariness about it. Any assignment of weights in lieu of order standings would be a conversion, but there is really just one logical way to do it. If first choice is given a weight of 1.0, the second choice must be somewhere in the range between 0 and 1, the midpoint of which is 1/2. Giving second choice a weight of 1/2, then allows us to go to third choice. Similar reasoning gives third choice a weight of 1/4, and we next go to 1/8, etc. Arbitrariness is removed when the choices are weighted according to the sequence 1, 1/2, 1/4, etc., and then normalized. For example, using 1 as the normalizing constant, three choices would have weights 4/7, 2/7, and 1/7. This seems quite constraining, but it is the best we can do with ordinal preferences.

The presence of the two bargaining solutions in the marriage problem bothers some people--maybe a lot of people when the method is applied to large matching problems like putting military graduates into their first duty assignments or medical school graduates into their residencies. Participants on the accepting side of the bargaining table are able to obtain more-preferred partners--on the average, but not always--than when they are on the proposing side. The preference raw data disclose that some participants are more popular than others, and it is reasonable to expect that the more popular ones should have their partners assigned first. The contest model bases its solution upon this idea and uses no bargaining.

Multiple Spaces and Adjacent Placement

Two special situations with matching deserve discussion at this point, and I will call them 'medical school problems'. They are related to the facts that (1) a medical school generally has many spaces to fill and (2) some students need to be placed together (married couples). This model works well here because it generates pairings in sequence, as described above. The schools simply put their spaces up for filling one at a time and keep replacing them until they are all taken. The student's preference is for the school, not the space. The married-couple situation is treated by having both partners give identical preference distribution functions on the schools, and zero preference for any school with only one space. The schools are asked to give identical preferences for both marriage partners. The preference of a married student for a school is programmed to drop to zero during the matching process when only one space is left there. When the matching program picks up a married student for matching, the second member of the couple is automatically sent to the same school by the next step of the program. I do not mean to imply here that all of the special complexities of medical school placement are easily solved with this model. There are others, such as placing a married couple into different specialties at the same school or switching spaces between specialties at a school.

Application to Voting

Finally, yet another variant of the roommates problem is solvable with this contest model: an election with three or more candidates. This problem has been a puzzle for at least 200 years. Condorcet (1743-1794) has his name attached to one of the main ways of treating it; another main way has been named after Borda. Both violate certain logical requirements in certain cases, as do all other methods proposed to date. (The so-called impossibility theorem of Kenneth Arrow addresses that--and his Nobel Prize in economics is for that theorem.) I would state, as a general principle, that no method of analyzing votes is correct if an illogical example (however rare in practice) can be constructed for it. Conversely, any method must be considered correct until an illogical example is found for it. (Unfortunately, the evaluation of vote-processing methods is so complex that it must rely mainly upon examples.) The voting problem is handled with this model by putting the voters into groups, each group composed of voters with identical preferences for the candidates. The groupability feature of contest analysis is utilized here. Then the candidates are given preference numbers, too, with a candidate preferring a voter with the same number that the voter used in preferring the

candidate. These preference data make up the scoreboard for the election, and the candidates are ranked from winner on down in the order of their applied strengths. *No illogical behavior will be present.*

The present situation with voting on three or more candidates is characterized as a breakdown in vote aggregation, with no method free of problems. This contest model eliminates the breakdown by eliminating the distinction between results obtained with or without aggregation. When it is computationally unwieldy to handle the voters separately, those that are identical in their voting pattern can be grouped--*with no difference in the analysis results.* This amounts to an aggregation of voters, not votes, and the impossibility theorem is bypassed. Even with grouping of voters, the number of groups can become large if the voters are allowed too much flexibility in their preference distribution functions. A hundred groups is routine in combat analysis, and the practical limit could be far higher, but it will be necessary to examine the combinatorial aspects of any large election ahead of time to be sure that the number of groups (including single-voter groups) is manageable. For example, a 10-candidate election with voters given three preferences to select (say with weights of 3, 2, and 1 for their first, second, and third preferences) would have $(10) \cdot (9) \cdot (8)$ or 720 possible groups of voters. Combinations can build up fast, but most groups would probably be unoccupied and grouping can be applied selectively (for the largest groups) to minimize the problem size.

The two-candidate voting case is not a duel. Besides the two candidates, there must be voter groups ranging in number between one and the total number of voters. (One group would be possible only for the unlikely case in which all voters split their preferences identically between the two candidates. If no voters were identical in their splitting of preferences, each voter would form his own single-voter group.) The winning candidate will be the one with the most votes, but the size of the win will depend upon how the voters split their preferences. The voters are sharing in the contest by having strengths of their own, and the details of this sharing will affect the strengths of the candidates.

Bills in legislatures are equivalent to candidates for office, in that they are voted upon. In analyzing the performance of a legislature, one would be interested mainly in the correlations among the legislators, just as one would be interested in correlations among voters and groups of voters after elections. All of this information comes naturally out of this contest model.

In voting, various options are now available for expressing preferences: order ranking, approval (yes or no), Borda counts, and cumulative votes. All of these have been shown to have logical defects in their processing with current methods. If any reliable processing method is ever found, it is surely not going to be applicable to all of these different preference schemes at once. I claim that this contest model, capable of operating only with normalized cumulative votes, is 100% reliable with respect to all of the major fairness and logic criteria applied to elections. To back up this claim, I have placed a discussion of vote-processing problems and their elimination via the contest model at another location on this Web site under the following link: [Voting Problems](#) .

Application to Game Theory

A recent development in contest analysis is the discovery that it will solve all problems in the important field of game theory, completely replacing current methods. A separate location on this site ([Game Theory](#)) is devoted to this topic. The reader should go there for a thorough discussion; only the bare outline is given here. The method entails (1) treating the game as a contest whose participants are *strategies* on one side and *outcomes* on the other, (2) converting all utilities in the game to preference distribution functions (players' preferences for outcomes), (3) computing vectors for the participants by means of the contest algorithm, (4) finding the vector sum of the participants' vectors and the projection of each individual vector upon the vector sum, (5) computing *qualities* of outcomes from products of vector projections, and (6) using these qualities to optimize the strategy probabilities. Every game has a unique and logical solution, and the method works for games with any number of players and any number of strategies per player, zero-sum or not.

Game theory has suffered for many years from the same sorts of difficulties that have plagued social choice: multiple solutions, sometimes no solution that makes sense, and no single way of solving all problems. Now, at last, contest analysis offers a neat, easily understood method to clean up the mess.

Final Remarks on the Philosophy

Why might one think that the normal way of processing an athletic scoreboard (or matrix of outcomes of individual games) has shortcomings? Don't we determine the winners of baseball pennant races on the percentage of games won, and without complaint from any quarter about the result? But what if the contestants had discernible differences, say in age, gender, or skill level? Or what if the games were badly distributed, with some players encountering mostly weak opponents? We do not know how to process contests like this so we try to compose our contests fairly, with matched competitors and equal play of one with another. But some contests occur without planning and with grossly mismatched contestants. It is precisely the differences in contestants, possibly invisible and in the form of special skills and attributes, which give rise to contest scores. We should be determining the strengths of the contestants instead of focusing on the scores they produce. With the model of this paper, we can do that, even for contests in which the differences among contestants are enormous. We could think in terms of mixed male-female competitions with the male and female winners computed separately. One more remark here about athletics is that a judged competition (as in diving or figure skating) is really a voting process, with the strengths of the judges to be determined as part of the process.

Alan E. Johnsrud, PhD

Arlington, Virginia, USA

1 June 1998 (Last revision, 30 November 2011)

E-mail: aejohns@erols.com

Some Worked Examples

To help clarify the ideas in this document, I prepared some small illustrative examples of a few applications. These can be found at [Examples](#). The solution to a larger problem--the entire '99-'00 and '00-'01 sessions of the U.S. Supreme Court--is reported at [Supreme Court](#). Other large problems are related to figure skating in the Olympic Games and to picking teams for the football bowl games. (Use the "topics" link below for the full selection of discussions on this site.)

Some References

The matching problem: [Roth's information](#). The voting problem: [Review of past efforts to cope with voting problems](#)

Display and Computing

Mathematical symbols can be misinterpreted by some browsers and displayed as false characters. It is for this reason that I have put this paper into PDF format instead of the normal HTML.

Placed on this site (at [Code](#)) is a copy of a simple QBASIC program which will analyze a simple contest with 6 participants. Unfortunately QBASIC is getting hard to run on computers with the latest operating systems. With some tricks, I can run it on an XP computer, and I can help those interested in this, if they will contact me. I can also send copies of QBASIC to those without it—and QBASIC programs for voting, matching, and game-theory problems.

There is now an adaptation of QBASIC called FreeBASIC which will run on 64-bit operating systems (which QBASIC will not). It requires some minor code changes to the QBASIC program.

Go to Home page: [Home](#) Go to Topics page: [Topics](#)