ROYAL INSTITUTE OF TECHNOLOGY

Sport-Sort

Sorting Algorithms and sport tournaments

Hans Block
Royal Institute of Technology
Sweden

Abstract

Many types of tournament schemes are used in different sports. We want to construct a scheme which gives the whole ranking order when pairs of athletes or teams compete. The scheme should be efficient, fair, and thrilling. The problem can be formulated as parallel sorting. The proposed algorithm will, according to simulations, take approximately 2.4 lg n rounds in a tournament with n participants. The computations will take $O(n^2 \cdot \log n)$ steps. However, both faster and slower special cases exist. Some limits for the number of rounds are given for variants of the algorithm. The scheme has been used with good results in real tournaments with up to 40 participants. At last we discuss in which branches of sports this algorithm will be most useful.

Tournaments and sorting

How it began

Back in 1948, I was 8 years old. The first Olympic Games after World War II were held. My two elder brothers and I played games. I always got the bronze medal. My eldest brother explained: The second best could lose in the first round. The silver medal can go to the wrong person. The cup is not always fair. The problem of fast, fair tournaments was stated.

Later, when my son was of the same age he played soccer. His team was not very good. Often, he played in groups of four teams. Three matches were done, but then he had to go home. What could an active father do to help his son? I invented a tournament system.

In the next generation: ***Give everybody a chance!***

The problem is older than so. Lewis Carrol, mathematician and author of *Alice in Wonderland*, wanted fair tennis tournaments. His problem: Make a tournament

with as few matches as possible which gives the three best players correctly.

Playing in pairs

There are many systems in sports where athletes or teams compare in pairs: In a series everybody plays with everybody. It takes long time. The cup is fast and gives the winner, but no one more, in a fair way. In soccer, the teams compete in a system with many series in several divisions. A team in volleyball can lose a match and still get one more chance. Tennis and chess have ranking systems, updated according to fresh results. Chess also uses the Monrad system, which resembles my proposal.

No solution is the best one. We want a system with

- Many parallel matches
- The whole ranking list
- Large tournaments
- Maximal effort required of all participants all the time
- Thrilling and unpredictable results
- Broad applications
- Simplicity
- Let weak players play!

Sorting by comparisons

Parallel sorting is an important problem in computer science. Many processors should be used efficiently. Comparing elements in computers give definite results, but in sports loops may appear. A tournament system must avoid loops.

Given a sorting algorithm, the computer scientist wants to know the number of comparisons, or rounds of comparisons. He also must know the computation time and the memory requirements, even if that is not critical for most sport applications.

A mathematician can learn about an algorithm by *simulations*, executing many sortings in a computer, by *heuristics*, plausible reasoning without strict proofs, or by mathematical proofs. The theorems can give upper and lower bounds for the rounds in a tournament. It also might give estimates for all but very few tournaments.

Results

The parallel sorting problem is not completely resolved.

1990 Leighton and Plaxton described a method and proved that it took at most $7.44 \cdot \lg n$ rounds with very high probability. n is the number of participants. lg is the logarithm with base 2.

1986, I presented the algorithm described below. Simulations show that most cases take approximately $2.4 \cdot \lg n$ rounds.

These results should be compared to the information theoretical bound of $(2 \cdot \lg n) - 2$ rounds. (That is easy to prove: n teams could be ranked in $n!$ ways. k rounds of $n/2$ matches could give $2^{kn/2}$ different results, and Stirling's formula gives the assertion.)

The new method

Other algorithms for tournaments

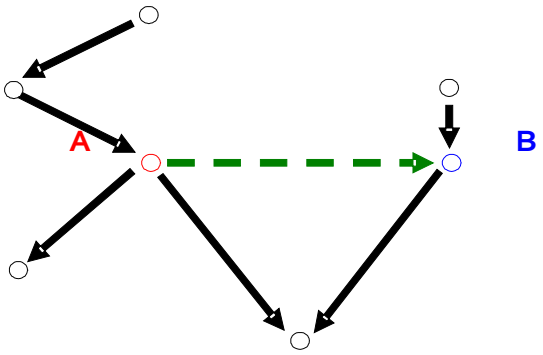
There are many ways to construct algorithms without loops. I made one called *simultaneous merging*. By proof, it takes in most cases $O((\lg n) \cdot \lg \lg n)$ rounds. However, the constant is large and the method is difficult to explain and implement. My next proposal is much better.

How the method works

If there is a chain of matches between A and E, so that A has won over B, B has won over C, ... and D has won over E, then we say that A is (directly or indirectly) **better** than E. We define **worse** correspondingly. We also count the **points** in a very special way:

Points (A) =
number of players **worse** than A –
number of players **better** than A.

Thus, the best player will get the highest score.



Points = Worse - Better

	Before		After	
	A	B	A	B
Worse	2	1	3	1
Better	2	1	2	4
Points	0	0	1	-3

An example of counting points. The arrows go from a winner to a loser. Before the round, *A* and *B* have equal points and play. We assume that *A* wins. After the match, *A*, can profit on *B*'s victories, and *B* is handicapped by the losses of *A*.

The players are placed in a preliminary ranking list which will be more and more correct as the tournament progresses. Before the first round, the **players are mixed** at random. In every round the following happens:

- **Matches** are assigned **between adjacent players** in the preliminary ranking list, if these players have not competed previously. In that case, some player must wait.
- The **matches are played**.
- The **points** of all players **are computed** as above.
- The **players are sorted** by the points. Players with equal points are mixed at random. This gives the ranking list of the next round.

The competition goes on until **all adjacent players have competed**. Then the order is correct.

No loops can occur!

We give a proof. Assume that no loops have occurred in rounds with number less than *k*. Then the points can be calculated, and the teams can be ordered by the points.

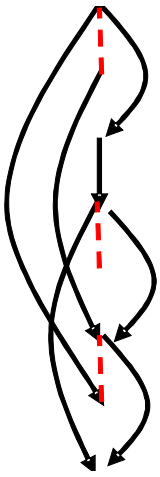


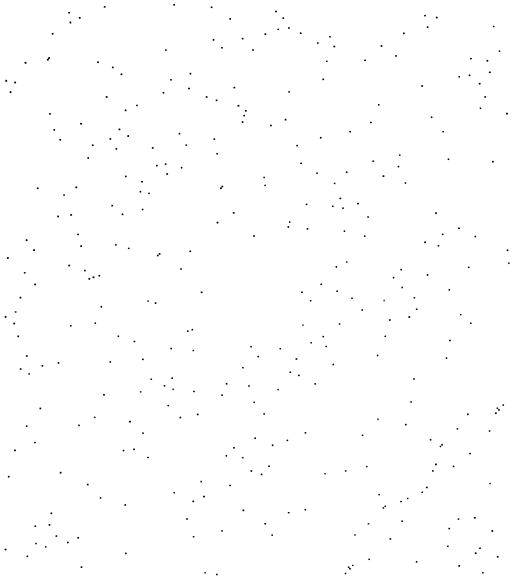
Figure. The teams are sorted by the points. If *A* has won over *B*, then *A* have more points, so all arrows point down. It is impossible to go more than one step up. Thus, there are no loops after round *k*.

An example

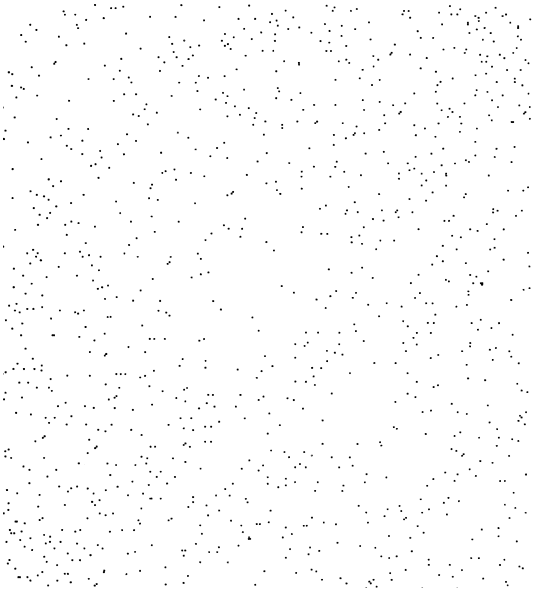
I made many simulations of this scheme. The teams were mixed randomly. We assumed that they had an inherent strength and always played according to it.

Given a tournament with 350 participants which took 20 rounds. We describe it with a sequence of bit matrices. The rows and columns are sorted by the players' strengths. A black point (*i*, *j*) means a relation between the players *i* and *j*. Before the first round, there are no black points.

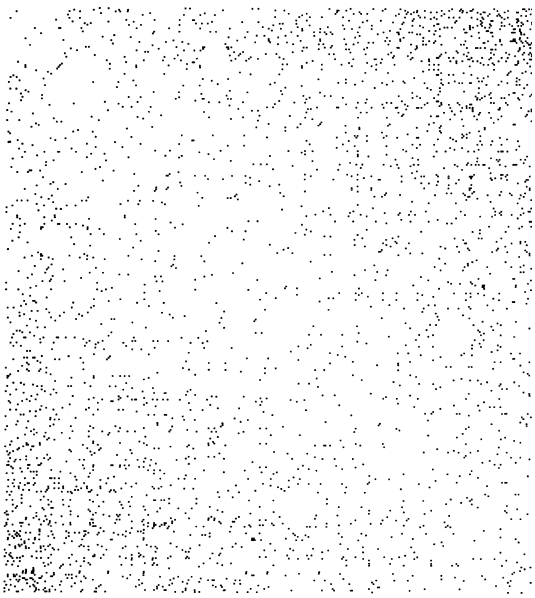
After round 1:



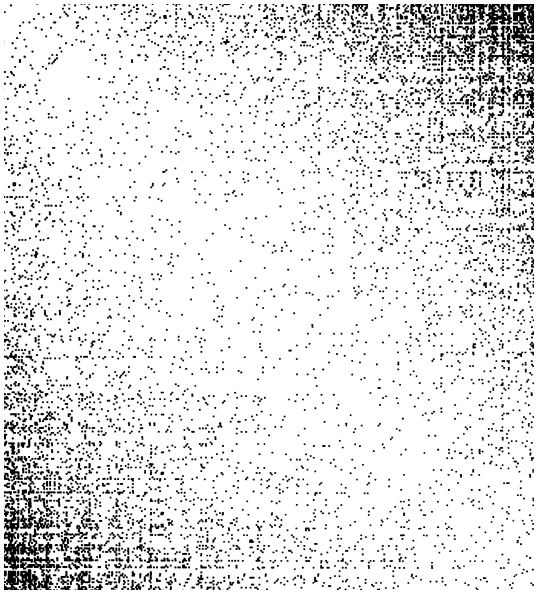
2



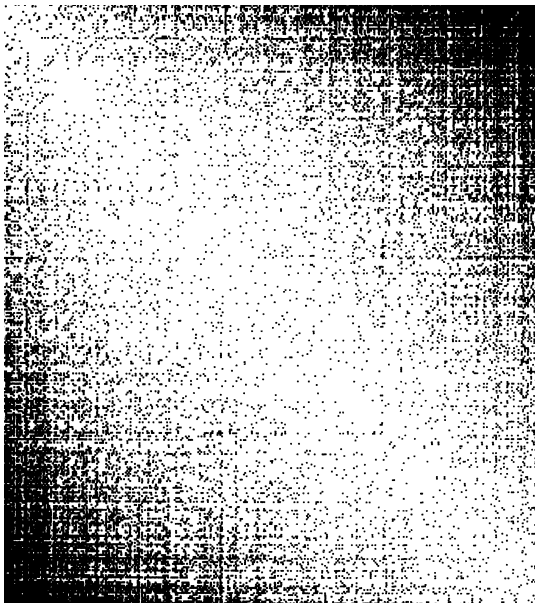
3



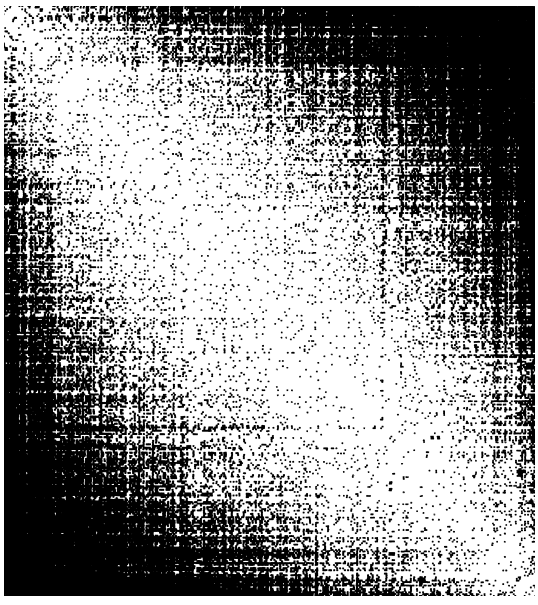
4



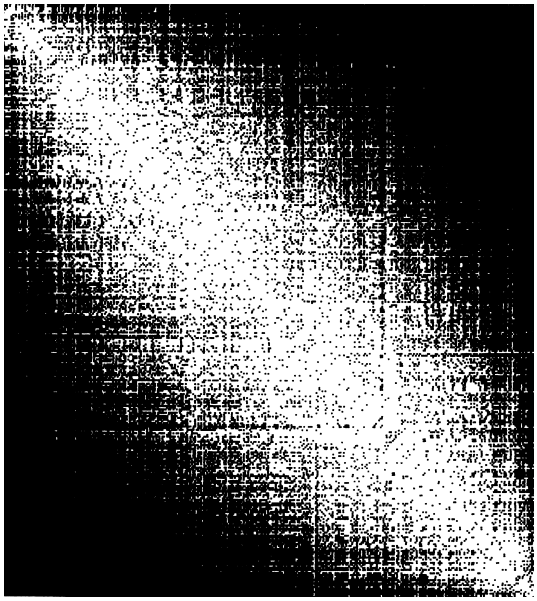
5



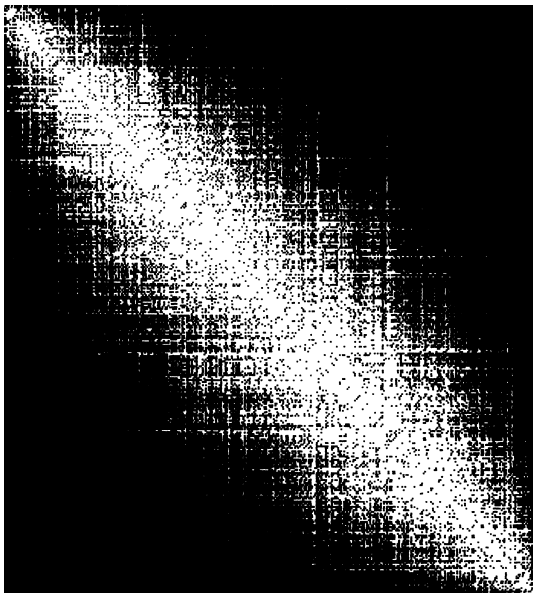
6



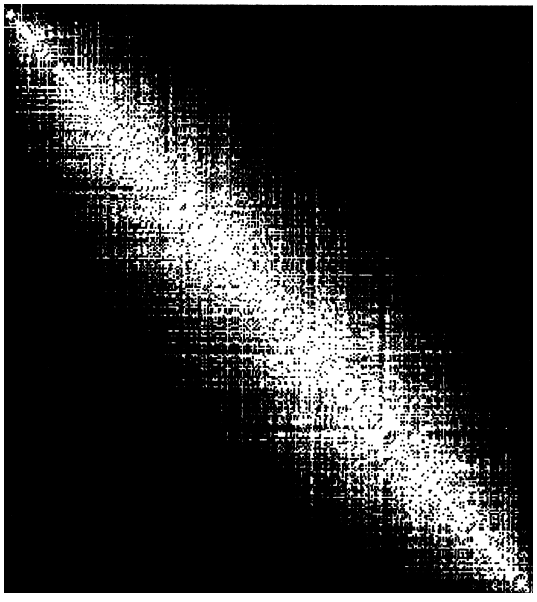
7



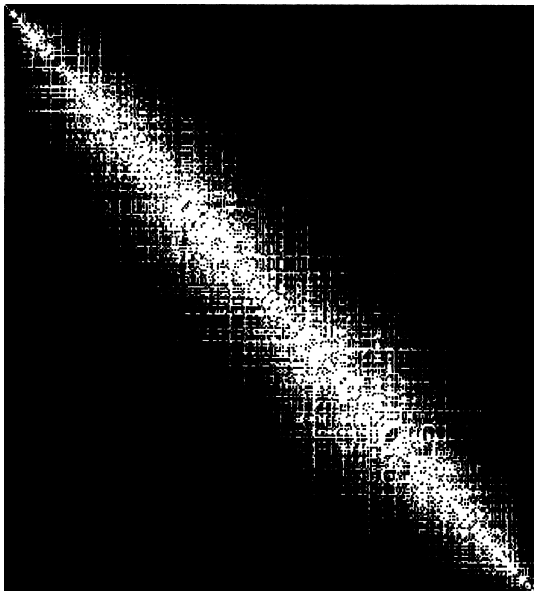
8



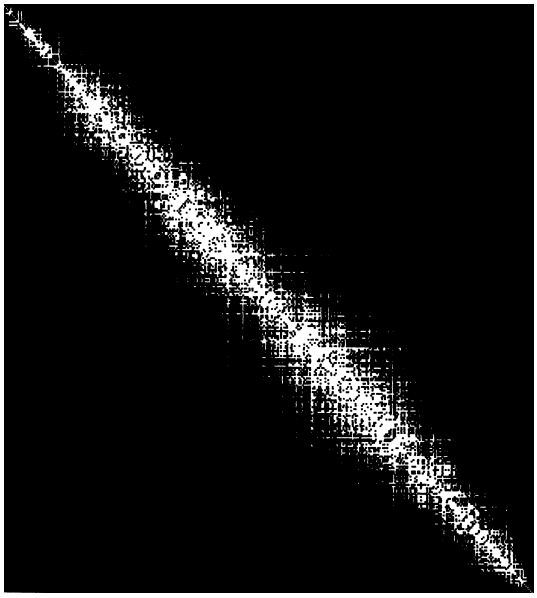
9



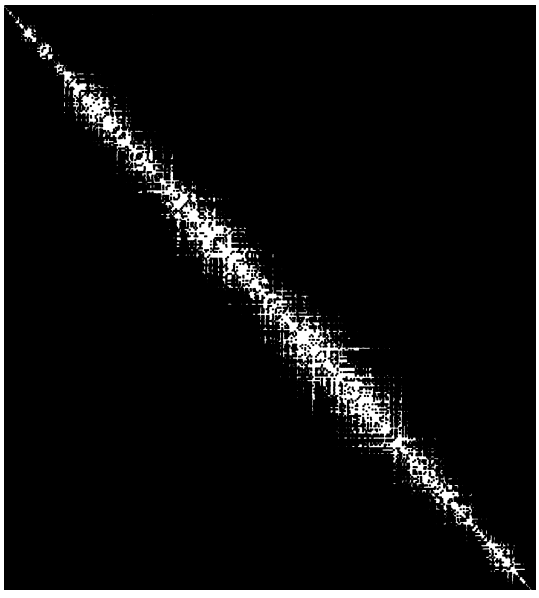
10



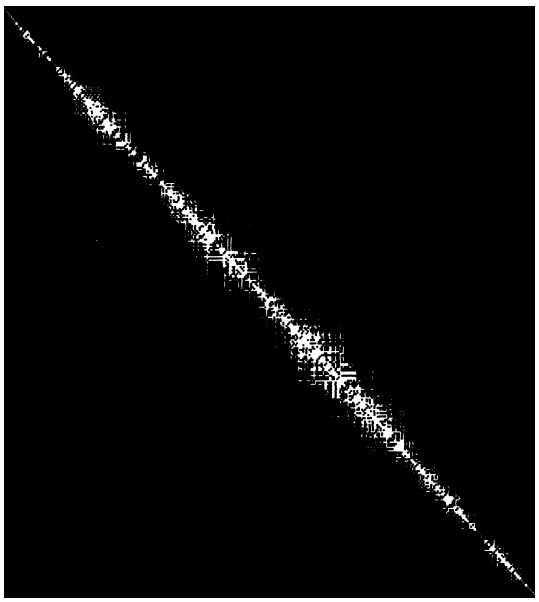
11



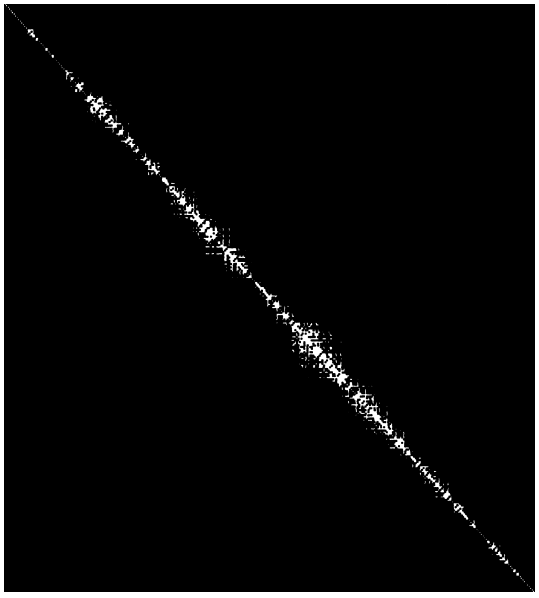
12



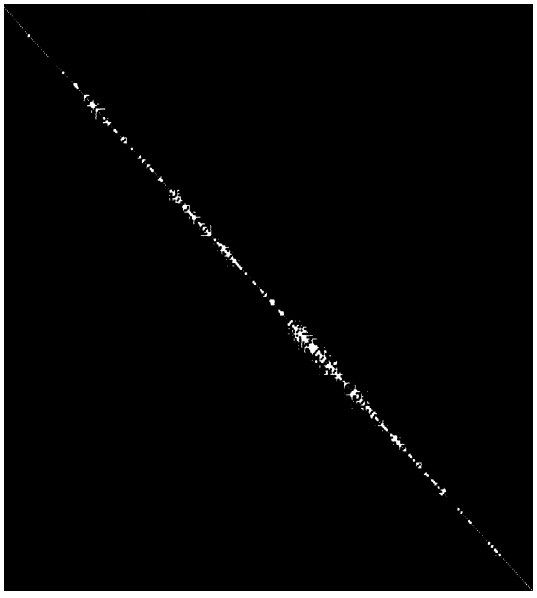
13



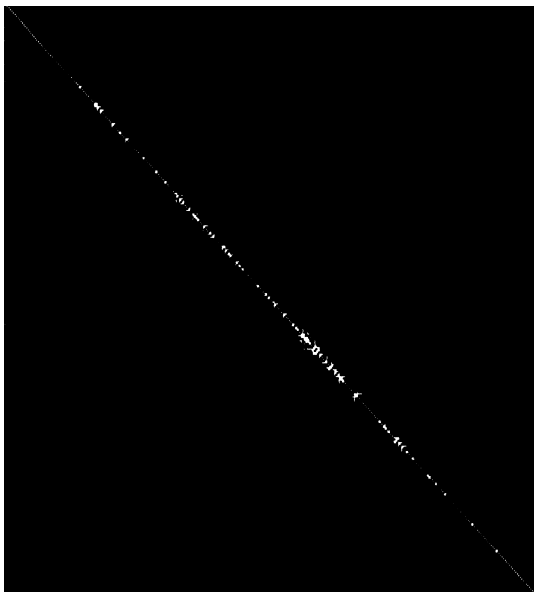
14



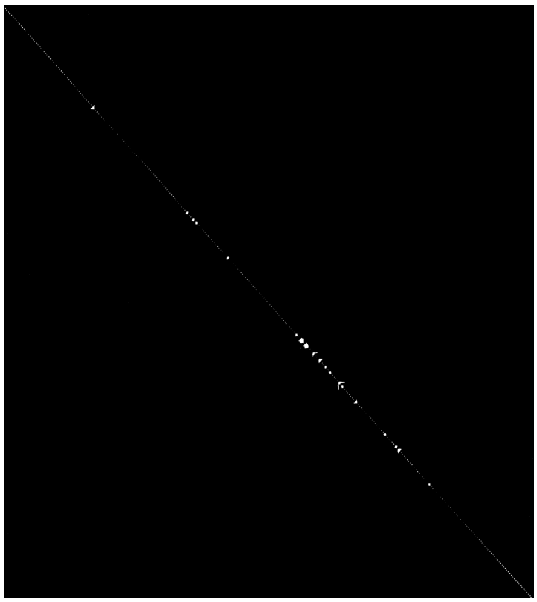
15



16



17



18

The tournament took 20 rounds, but the thin line of unknown relations is not visible in the printing of the last few rounds.

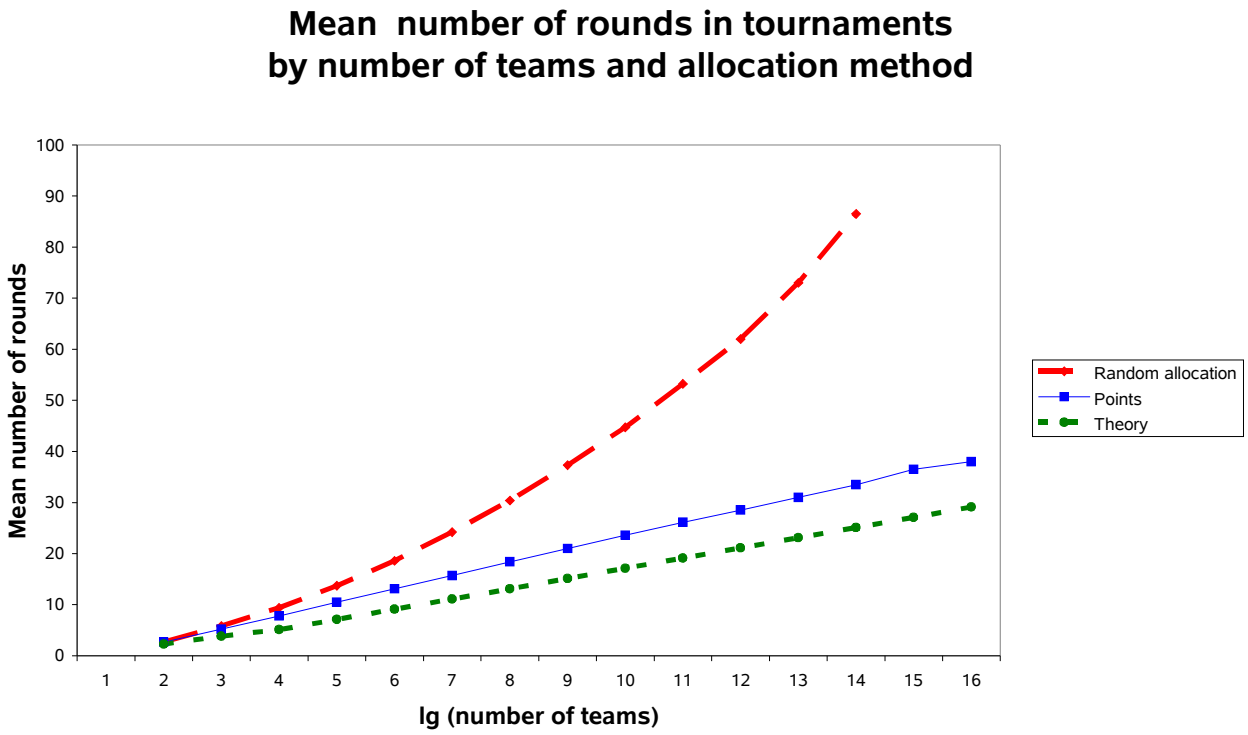
The figures indicate that number of known relations grows exponentially in the beginning and that the number of unknown relations decreases exponentially in the end. After lg n rounds, the ranking looks rather correct.

Typical behavior

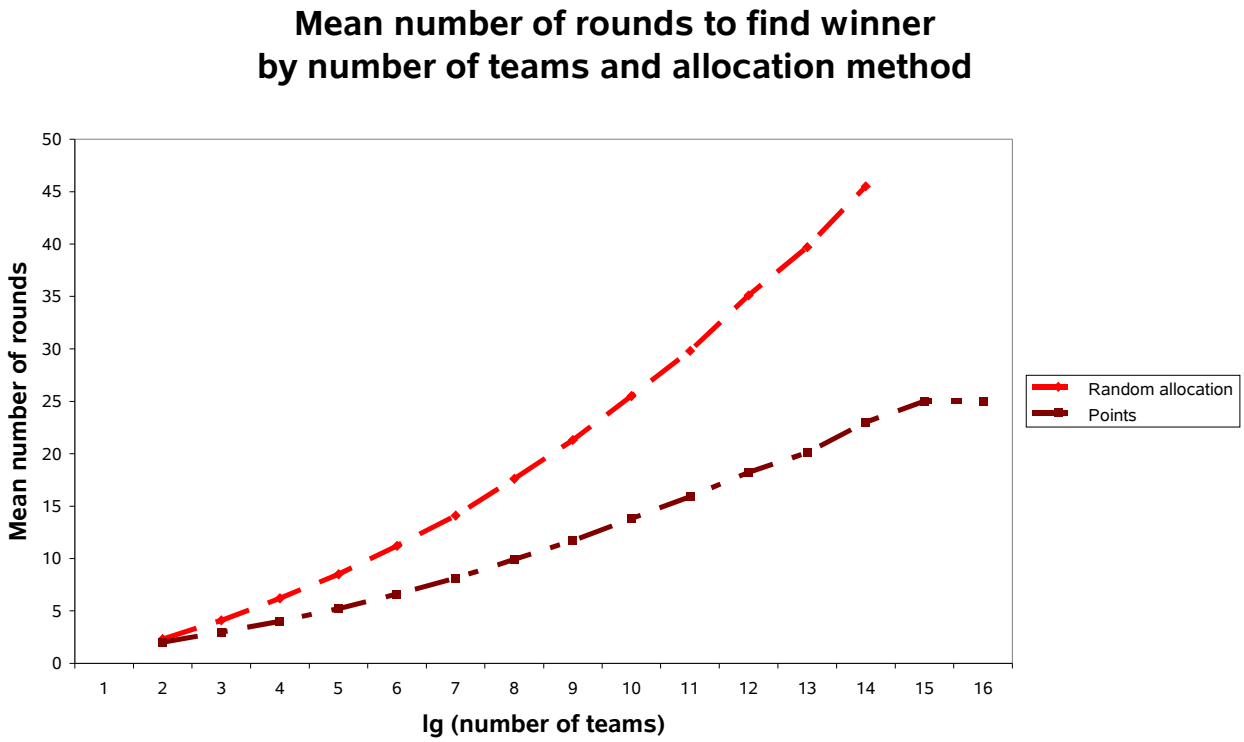
It is easy to guess that the algorithm is fast. The information from a match is used to its very limit. The matches are efficient, since the participants become more and more equal in strength.

There are some heuristic arguments that the time is logarithmic in n. The points are the sum of many small random terms. We could thus assume that the distribution of the points is approximately normal. The points determine the rank, so we can assume that the rank is also normally distributed. A rough model predicts that the standard deviation of the rank is reduced by a fix factor, which agrees with the experiments.

We also simulated many tournaments of different sizes and noted the number of rounds to give the winner and to complete the tournament. We did the same for a similar algorithm in which matches were assigned randomly between teams with no relations (in this case loops could occur!) We compared the results with the theoretical limit.

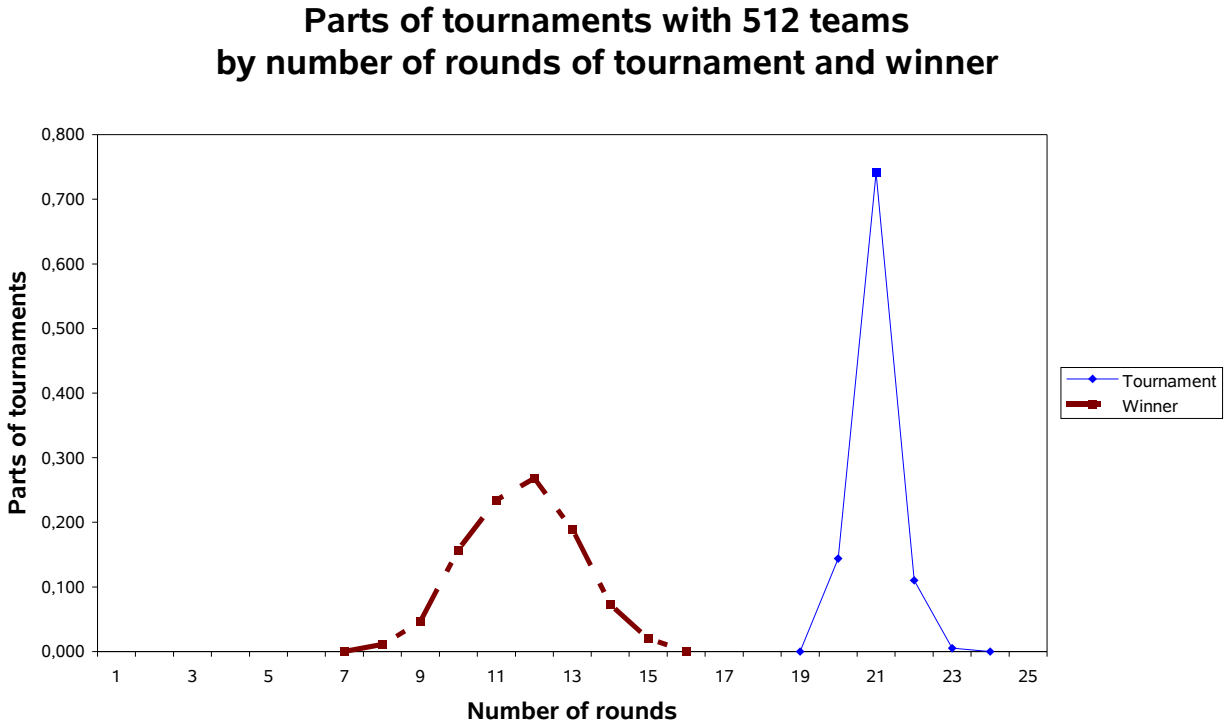


The efficiency of the method is obvious.



The result is somewhat disturbing: up to say 256 teams, the winner is ready after lg n rounds. Later on, it grows almost to 2 lg n rounds.

We also wanted to see the variation of the number of rounds.



The number of rounds for the whole tournament seems pretty constant. The distribution of the rounds of the winner seems more like a normal distribution. That would indeed indicate that large deviations may occur.

We also see that the winner is ready much before the whole ranking list.

Pathological cases

I have spent long time to find the properties of the algorithm. In order to get examples: allocate matches between teams with equal points in the most – or least – efficient way, and fix the outcome of the matches to get the desired results. At the end, the matches define an order of the participants, and this order could be found by the matches in the example.

I found many odd behaviors, e.g.

- A whole tournament took less than $\lg n$ rounds
- The winner was appointed in only $3 \cdot \lg \lg n$ rounds
- Half of all the relations were fixed after $4 + \lg \lg n$ rounds
- The winner played only 3 matches

Other examples showed that tournaments could be very slow:

- Winner might require $\approx 2 \lg n$ rounds
- Tournament might take $2.7 \lg n$ rounds

A very complicated example showed:

- Tournaments taking as much as $\frac{\lg^2 n}{16}$ rounds.

The last example is shocking from a theoretical standpoint. However, it has no practical interest. The estimate is worse than in the previous example only for tournaments with more than 10^{13} participants, more than 1000 times more than the world's population. Even with so many players, this result is extremely improbable.

What could be proved?

There are some definite results. The lower bound for the number of rounds is $\log^* n$, an extremely void result. On the other hand, a probabilistic variant of the algorithm takes at most $O(\lg^2 n)$ rounds, but with a rather large constant. The simulations indicate this bound is sharp.

The upper bound of the number of rounds is still an open problem.

There might be a way to prevent the algorithm's worst cases. They occur when we have "inbreeding", many matches within small groups of teams. Instead, we want to spread the results as rapidly as possible over the whole graph.

We have the freedom to reorder participants with equal points. For a given match between the players A and B, we can count how many new relations we win in the cases when A wins or A loses. The least number we call *guaranteed relations* for this match. If we allocate matches looking one step ahead in order to maximize the total number of guaranteed relations, then the most pathological cases are avoided and we might get better results.

Applications

Sport and the transitive model

The idea of transitivity is evidently not correct. The results vary much due to injuries or the players' mood of the day. The scheme uses the information of every single match for the ranking order. Errors in the input will cause great errors in the final order.

Players in e.g. table tennis may have different specialties, so loops may occur even when we use common, stable match results.

However, this should not prevent us from testing the tournament scheme. The reality is complicated. We can only use models to describe it, and the models might be more or less correct. Sport is full of surprises. The outcome is not always as expected and might seem unfair. Such uncertainty should also influence tournament schemes. With limited time, this system may be optimal.

Experiences

Some 10 tournaments have been made by this scheme: badminton, table-tennis, squash, volleyball, basketball, karate, and chess. There has been a clear interest from bowling, fencing, and tennis.

The method is possible to understand by participants and press. The competitions have given results which could well be expected.

In none of the tournaments there was time to get a full ranking list. Instead, the tournaments were stopped after about $1 + \lg n$ rounds. In order to keep the interest they were completed by cups with the four best participants.

Which sports?

Some conditions are favorable for Sport-Sort tournaments.

First, the scheme is most interesting for large tournaments. 10 participants may be a minimum.

Since every match is important, and there is no chance to repair a defeat, there should be a high probability that the stronger player wins. For example: In table tennis, there are many balls to win before the match is over. These balls could be regarded as independent trials, so the probability that the stronger wins is great. On the other hand: In chess it is quite possible that the stronger player loses a pawn early in the game. The variation of a game of chess could be greater than in match of table tennis.

There must be sufficiently many fields (courts, tables, boards) available so that the matches could be played at the same time.

The athletes must accept to play during the whole competition, even if they have failed and will get a poor placement. That may not be the case in e.g. professional chess championships.

A series takes $n^2 / 2$ matches, this method needs $2n \cdot \lg n$ matches, but a cup is ready after only n matches. The organizers must have a genuine interest to let also the weaker participants play, and the sport must have sufficiently many courts (tables, …) available.

It is good if the matches take a roughly fixed time, as in soccer, and not a highly variable time, as in tennis.

A good tournament

Every year there is a chess tournament for almost 1000 players (mostly amateurs, but also professionals) in Stockholm City Hall (were the Nobel Prize winners eat their dinner). The tournament takes two days. The first day people are playing in groups of 8. On the second day, most of the players have disappeared. Why not apply our scheme instead, to settle almost the whole ranking list?

Improvements

Match allocation

The algorithm allows the organizers to reorder the matches between players with equal points. This option could be used for several purposes:

- Avoid that athletes from the same club or city to compete in the first rounds
- Prevent players who have drawn to compete again
- Avoid that matches are allocated within a small set of players
- Maximize the number of guaranteed relations

These claims will not always be possible to fulfill. In that case, you must give the claims different weight and use standard methods for optimal weighted matching. However, the implementation is more difficult, and the organizers must state exactly how the want the tournament.

Logistics

Arranging tournaments gives many practical questions. How should the advantages be distributed, when we do not know the matches in advance?

In e.g. table tennis, one hall might not be sufficient for all players, so the tournament must use several halls, and the players must take buses at some intervals. Could the matches be restricted within a hall?

On other sports, e.g. badminton, there may be lack of courts in a hall. If the tournament should be ready in time, the courts must be used all the time. However, the players must know in advance that they should play and have time to warm up.

A game of chess can take many hours. After 40 moves the game is interrupted and taken up again the next day. This must not prevent other participants to play.

Our scheme must take these circumstances into consideration and still avoid loops. How this is be done will not be shown here.

In bowling all lanes are a bit different. In order to avoid unfair advantages, the players compete in pairs on the same lane. A player should not play more than once on every given lane. A tournament software for bowling must match not only player to each others but also lanes to players. This also requires algorithms for weighted matching.

Conclusions

Parallel sorting is an important and difficult problem in computer science. The present method has proved to be very fast in simulations. It deserves a study of the finer points. However, there exist several pathological cases. Not much is proved about the original algorithm. It is an open question if the algorithm could be changed to run in logarithmic time.

In practical tournaments in several sports the scheme has worked fine. Software exists, but needs updating.

Why not try?

Questions are answered by:

Hans Block
Monstringsvagen 126
SE-184 33 AKERSBERGA
SWEDEN
Phone: **+ 46 8 540 632 39**
Mobile: **+46 73 938 76 53**
E-mail **hans.block@swipnet.se**
Website: **home.swipnet.se/~w-35364/**