An Algebraic Functional Domain-Specific Language for Tournament Structures

Algebraic Graph Construction of Sorting-Network-like Structures

Mike Ledger, supervised by Ranald Clouston, at Australian National University

November 20 2023

Abstract

An Embedded Domain-Specific Language (commonly known as eDSL) is developed for relatively easy expression and analysis of arbitrary tournament structures. The core tournament structure type inspired by work on algebraic graph representations in Haskell, and sorting networks, enabling composition in terms of either interleaving or sequencing. The core design balances between allowing arbitrary logic within a tournament structure and what static analysis can be achieved given such a structure. Metrics such as length, fairness, resource use, and reliability, are analysed on a variety of tournament structures. Similarity between sorting algorithms - in particular sorting networks - is observed and common tournament structures are contrasted with sorting networks to see how viable they can be as tournament structures. A tool is also developed to perform and visualise tournaments described within the eDSL.

Contents

1 Introduction

Tournament structures are used to determine a ranking of players in a game, such as a real-life sport or an eSport. Many different tournament structures are employed in real games, for a variety of reasons; for instance length, fairness, resources available, are all determining factors. The choice of what tournament structure is most appropriate for a given set of circumstance is mostly an open one, and many tournaments are ran as a composition of other tournament types.

There is then a possible contribution to be made in a software tool that would help design and evaluate tournament structures. This project encapsulates an effort to do that over the course of my undergraduate Advanced Computing Project (COMP4560), by creating a Embedded Domain-Specific Language for tournament description and evaluation, as a Haskell library. The software library developed for this project is available online on GitHub, which also hosts API documentation.

2 Background

The economics of sports and eSports tournaments are also worth noting here. The social impact of this is out of scope for this project, but clearly with [XXX: growth in Twitch/stream viewerships] these tournaments are becoming a significant part of the zeitgeist.

3 Old Background

4 Old Aims

1. To create a computer language that enables the description of common tournament structures, and also to enable creation of new tournament structures as compositions of existing ones.

2. To create an interpreter for this language that can evaluate tournament descriptions; to be able to run the tournaments described by the language defined in Goal 1.

3. To systematically evaluate the efficacy of different tournament structures in terms of quantifiable properties such as fairness, length, repeat-avoidance, and so on.

These aims are underwritten by my own experience as a tournament organiser and developer of a popular website that facilitates the running of eSport tournaments; in running these tournaments, questions of fairness and the ideal arrangement of tournament brackets are commonplace; validating existing methods or otherwise creating newer and more effective ones would be a valuable development here. [1]

A key observation of this project is the existence of an equivalent sorting network to many tournament structures. In particular, any tournament structure whose rounds do not depend on the results gathered from prior rounds, and whose matches can be expressed in Compare-and-Swap Matches (CASM), have an equivalent sorting network construction. A hierarchy of match types is then constructed, and it is shown that CASM is a specialisation of Points Matches (PM), from which we can also construct tournaments based on ranking systems such as Elo ((cite:@elo)). These are common

This fact that a tournament can be constructed from any comparison-based sorting algorithm (swapping comparison with a game/match[2]) is an important observation that I have made and informs the aims of the project.

The research I will be conducting has these aims:

[1] In my experience, the question of what tournament structure is most effective comes up very frequently, and players often do feel their placement in a tournament was unfair, or a player's seeding unduly caused them to place higher or lower than they "should" have according to an assumed skill level. I believe avoiding these scenarios would improve player satisifaction and enjoyment.

[2] **Game** shall be used to refer to a contest between two players from here on. The exact terminology is not that important.

1. Creation of a domain-specific language (DSL) to describe tournament structures
2. Use of this DSL to describe common and novel tournament structures:
   (a) Round-robin
   (b) Elimination/knockout
   (c) Double/triple/... elimination tournament
   (d) Swiss-style tournament
   (e) Arbitrary sorting network designs, such as insertion sort, bubble sort, bitonic sort, etc.
   (f) Partial sorting algorithms or partitioning algorithms such as quickselect or median of medians method
3. Use of this DSL to describe compositions of created tournament structures:
   (a) Sequences of tournament structures (as in a "multi-stage" tournament)
   (b) Partitions of tournament structures by Divisions, where a each player in a given Division D_1 is expected to be of a higher skill than each player in the next division D_2, and so on.
   (c) Groups of tournaments, where each group is expected to have the same average skill level as each other.
4. Analysis of these tournament structures according to quantifiable metrics, through simulation. The simulation can take place using generated skill distributions (most likely through Elo [3]), or by using real-world data such as Elo distributions of Chess or Tennis players, which is publicly available data.
   - Efficiency
     - The time it takes to complete a tournament
     - The number of games that can be ran in parallel
   - Fairness
     - repeat avoidance or not
     - Preservation of pre-determined skill levels
     - Maximum waiting/idle times for players between games
   - Robustness against manipulation - can strategically losing games improve a player's final ranking? e.g., can strategically losing games in an early round ensure an easier later tournament?
   - Robustness against inaccurate seeding (where the true skill level of a player does not coincide with their seeding [4] - such as if they simply have a "bad day", or their skill level was perceived as higher or lower than it truly is by an event organiser.)

[3] A common player ranking system, designed originally for Chess, that allows for computing the probability of a player winning against another.

[4] The rank a player is assigned at the outset of a tournament. This is especially important in elimination-style tournaments. The worst-case scenario helps illustrate why: If the player with the true highest skill and the player with the second true highest skill are seeded to play each other in the first round, either one is guaranteed to be immediately eliminated.

    - Ranking precision
    - Reliability - how repeatable is the result of a tournament?

Given these aims, it is important to consider the existing literature on topics. These are the topics I am looking for roughly. I use "efficacy" here to encompass fairness, reliability, etc., as outlined above.

- Efficacy of tournament structures
    - Simulations of tournament structures under real-world or generated skill distributions
    - Improvements that can be made to existing tournament structures
    - Common pitfalls of existing tournament structures
    - Relative strengths or weaknesses of tournament structures
- Studies on the efficacy of past real-world tournaments
- Domain-specific language design and development
- Existing software tooling that can help accomplish the aims above

Things out of scope of the research aims, but that may be useful to note, include:

- Psychology of sports and eSports tournaments
- Effect of different tournament structures on player performance
- Player preference to different tournament structures
- Perceived (by players) fairness of different tournament structures

5 Literature Review

I proceed here to review XXX papers relevant to the above topics. Each section is the title of a paper and links to a proper reference in the bibliography. It is worth noting that this section overlaps with the notes I have written already for the project in my GitHub repository for it, as it is my current COMP4560 project. [5].

Each paper has its own measure for what constitutes fairness, balance, competitive development, etc., which are usually similar but not necessarily identical. My research project therefore aims to provide a tool that makes it easy to simulate tournaments and to measure such metrics in a unified fashion.

[5]These notes are available online, and so the similarity will hopefully be noted by TurnItIn on submission of this literature review

5.1 “The structure, efficacy, and manipulation of double-elimination tournaments” [1]

This paper provides information about double-elimination tournaments in particular, and in contrast to single-elimination tournaments. Several important theorems are provided as well as statistical analysis performed on the efficacy of tournaments.

Statistical analysis is performed to compare the reliability of single-elimination tournaments to double-elimination ones, where double-elimination is shown to be much more efficacious in allowing the most skilled player to win than single-elimination. Simulations are performed using chosen models for the probabilities of players winning against each other, rather than on real-world data.

Manipulation of double-elimination tournaments is also considered, and an interesting case study provided to demonstrate the need for tournaments that are robust against manipulation: “in the 2012 Olympics, four of the top badminton teams were disqualified for trying to intentionally lose matches, causing an uproar and angering fans. While the tournament structure used there was not a DET, this demonstrates that players really will exploit poor tournament design when possible.” The importance of seeding in the outcome of elimination tournaments is noted as well. Several theorems are provided on the complexity of manipulation of a tournament by players.

Of note is that this paper provides a result that shows that a double-elimination Link Function. The Link Function is the algorithm that chooses where in the lower bracket a player from the upper bracket should go after a loss. The choice of Link Function is quite important in order to avoid re-matching players who already faced each other in the upper bracket, for as long as possible. It is shown that steps taken toward repeat avoidance need only be done up to $\log(R)$ with R the total number of rounds, and an algorithm is provided as a suggested “optimal” Link Function using the provided primitives (Swap and Reverse) for constructing link functions. I have implemented this suggested algorithm in Rust in my tournament website `https://kuachi.gg`; the implementation is available online.

5.2 “Double-Elimination Tournaments: Counting and Calculating” [2]

This paper provides broad information about the construction of Double Elimination tournaments. The efficacy of “unbalanced” double-elimination tournaments is considered in detail. A system for uniquely numbering single-elimination tournaments is also provided, with extension then to number double-elimination tournaments by the structure of the lower bracket as well as the linking function used.

Statistical analysis is performed by using an assumed “preference matrix”, denoting the pairwise probabilities of one team winning a game against another. This is an interesting approach that may be extremely difficult to compute for larger tournaments (only 4 player tournaments are considered by preference matrix), but offers several advantages over “traditional” ranking methods such as Elo. In Elo, all players are assumed to have an absolute quantifiable skill level, that satisfies transitivity; if player A is more skilled than player B, and player B is more skilled than player C, then A must be more skilled than player C. A preference matrix approach allows for the fact that some players may do particularly well or poorly against other players. It may be possible to calculate a preference matrix from existing public data from existing games, by assigning a secondary ranking to players by treating each possible pair as its own separate game.

The larger double elimination tournament shown in this paper does not to have a "balanced" lower bracket. Convention in modern double-elimination tournaments is that, to maximise fairness and minimise the number of rounds required, one should alternate between rounds where players are from the lower bracket play against each other, and where "new" players are added in to the lower bracket from a round in the upper bracket. This is shown in [1].

5.3 "Simulating competitiveness and precision in a tournament structure: a reaper tournament" [3] and "Reaper Tournament System" [4]

I consider a pair of papers here sharing two of the same authors; [4] describes most of the results and [3] develops the knowledge of the Reaper tournament system further, and creates a similar (but new) tournament structure called Reaper elimination.

This paper proposes a novel tournament structure called a "Reaper" tournament. It has several advantages to existing tournament structures, that are outlined throughout the paper. This tournament structure is interesting as it is the sort of structure that I would like to enable the creation and analysis of through the DSL.

The structure of a Reaper tournament is not intuitive to me, but I repeat it here in my own words in order to help my understanding of it that it operates as an inverted single-elimination tournament initially, where only losers "advance", and from there a unique algorithm for repeatedly selecting and eliminating the worst player is applied. This seems to have similarity to a selection sorting algorithm. Because the Reaper tournament system is a complete sorting algorithm, it has 100% ranking precision.

The number of matches required in a Reaper tournament is not given a general formula in the system, nor the number of rounds, which is a significant weakness to its adoption as a tournament structure in practice - events need to happen usually within known time constraints. Description of the Reaper tournament system as a sorting network may help to elucidate its properties. For $n = 8$, the Reaper tournament requires $m \in [15, 17]$ matches compared to $m = 14$ for double elimination or $m = 28$ for a round-robin.

It is also shown that the stability progression, measuring whether winning a game is more desirable than losing, is preserved in the Reaper tournament structure. It is never a desirable outcome to lose a match in the Reaper tournament structure.

5.4 Description of the Reaper tournament structure algorithm

I reproduce in my own words the algorithm for the Reaper tournament structure here.

Information:

- Each player has a respect list of players who they have previously lost to. This is updated every time a game occurs.
- The tournament is assumed to be $n = 2^k$ in size; there must be a power-of-2 number of players.

Steps:

1. Reaper selection: In Round 1, pairs of players are matched together, so that every player is in a match. The losers in the round are then paired against each other, and again, until a round where only a single player loses a match (who lost all matches prior to this round), and they are eliminated from the tournament. Let the winner of the final game in this step be the Reaper.

   This basically describes an "inverted" single elimination tournament - where to proceed to the next round, you must lose the current round. The "winner" (i.e., loser of all games) then of this inverted single elimination tournament is the one who is actually eliminated from the tournament.

   The question of what matching algorithm is used is left open by the authors of the paper, but it is likely significant in determining the outcome of the Reaper selection stage.

2. Reaper candidates: A candidate list is created consisting of:

   - If there are players who are not in a respect list, those players.
   - Otherwise, the players who are in the respect list of the Reaper.

   The size of the candidate list then determines the next step:

   - If > 1, proceed to (3).
   - If $= 1$, proceed to (4).
   - Otherwise ($= 0$), the tournament ends.

3. Candidates match: The two best players play each other. Update the respect lists accordingly and go back to step (2).

4. Reaper match: The single player in the candidates list plays the Reaper. The loser here is eliminated and is ranked above the previously eliminated participant, while the winner is set to be the new Reaper.

5.5 Reaper Elimination

A new structure is proposed in the following paper [4] that develops the Reaper tournament structure to give it an upper bound on the number of matches required, and a static tree structure. A visualisation of the Reaper elimination tournament structure is provided in that demonstrates a static structure to the tournament. Thus, it is a tournament that could be expressed as a sorting network. It is shown that the number of matches required is $O(N \log_2 N)$.

The second paper analyses two-stage tournament systems where a group stage precedes an elimination stage. The group stage has multiple groups of players in each group, and a tournament structure such as round-robin (or Reaper), is conducted within that group.

Various metrics are created to measure the efficacy of tournaments in practise and in simulation. The key metrics are ENM, meaning "expected number of matches", ARW, the "average rank of the tournament winner" (ideally, 1), and RankCor $\in [0, 1]$ where a value of 0 means the tournament had a completely random result with respect to the player's "true" skills/rankings, and a value of 1 means that the tournament perfectly preserved those a priori rankings.

Theoretical experiments for on 8 player tournaments are conducted that show the excellent RankCor of the original Reaper tournament structure. Double-elimination stages are also shown to have quite good RankCor (at this size of tournament). Real-world tournament data is also used that demonstrates the robustness of double-elimination tournaments in terms of RankCor, with Reaper tournaments also performing excellently for up to double the number of matches required.

5.6 "Quantifying the unfairness of the 2018 FIFO World Cup qualification" [5] and "Risk of Collusion: Will Groups of 3 Ruin the FIFA World Cup?" [6]

These papers look at real-world sports tournaments, namely the FIFA series of soccer[6] tournaments. As these are huge events with massive prize pools and carry great prestige for participating teams, nations, and hosts, examination of these events for fairness criteria is important. These papers demonstrate how real-world data can be used to examine and quantify fairness of tournament structures.

It is shown in [5] that the origin continent of a team has an outsized effect on the likelihood of a team in qualifying into the FIFA World Cup in 2018. The key takeaway is that a fixed draw rather than a random draw for qualification would reduce unfairness. Unfairness is measured by "[...] ranking the teams according to their Elo, and summing the differences of qualifying probabilities that do not fall into line with this ranking". This unfairness metric may be useful in this research project in examination of the fairness or not of arbitrary tournament structures.

In [6], the conditions required to aggravate the risk of collusion between teams is examined. This can occur when two teams in a Group stage are already guaranteed entry into the proceeding stage, but the result of their match can adversely affect whether or not another team in that group makes it through to the next stage or not. Examples of collusion are examined in real-world games. Soccer seems especially susceptible to colluding outcomes as draws are possible outcomes in the sport; teams may agree in advance to draw against each other, and neither will lose face nor prestige, but both may then be enabled to gain points required to proceed to the next stage of the tournament. Such examples seem quite common. The risks of collusion are assigned probabilities and examined in detail. Situations are examined where a team is happy to lose by a small amount, and play to achieve that result.

[6]Football?

5.7 "Handling fairness issues in time-relaxed tournaments with availability constraints" [7]

This paper examines computational complexity of time-relaxed tournament game scheduling. That is, the problem of scheduling games where there is not a tight deadline to complete the games, but there may be sporadic player and venue availability. This situation frequently occurs during "long format" group stage formats which are ran over weeks or months, where the scheduling of each game is done by each player participating in that game together. However, this is out of scope to the research aims of this project. The fairness measures proposed by this paper also concern scheduling, which is outside of the scope of this project.

5.8 "The impossibility of a perfect tournament" [8]

This paper provides an important result that shows that their constructed fairness and balance metrics trade off against one-another, and elimination tournaments cannot be constructed that maximise both metrics.

The fairness metric here is that the sum of the ranks of winners of each match must be maximised across the whole tournament. This is an interesting definition that intuitively works quite well when the tournament structure is also minimising the number of matches required - one could construct a degenerate case tournament structure that maximises this sum, by, for example, matching 2 weak players repeatedly until the sum generated by the winners of those matches must be greater than the sum generated by the winners of the other matches in the tournament. Of course, that would not be an elimination tournament.

The balance metric here is to minimise the difference in ranks between players across all matches. By doing this, you create tournament structures that provide as more information about players who are closely matched. In the single elimination case, it is clear that maximising balance minimises fairness. Maximising balance can have the effect of increasing spectator interest, as closer games are assumed to be more exciting to watch than "blow-out" games, which I can validate from my own anecdotal experience.

The paper proves that a directed tournament that contains any sub-tournament where 4 unique players play 2 games in the same round, then the tournament cannot minimise both the fairness and balance metrics. The terminology for maximising and minimising fairness and balance is somewhat confusing in the paper, as, for example, the section "Tournaments that Maximize Both Fairness and Balance" discusses a tournament structure that in fact minimises fairness and balance, and makes no mention of tournaments that maximise it. Taking the proof at face value, that it shows that the balance and fairness metrics cannot be minimised (with the necessary conditions satisfied), this does not seem to preclude the design of a tournament structure that maximises the fairness and balance metrics as the title of the paper would imply. This may just be my limited understanding of this paper; a better reading may be required. In that section, the paper provides an interesting 4-player tournament structure similar to double-elimination, which does minimise (i.e., "make bad") the fairness and balance metrics.

The author concludes that a perfect tournament design cannot be made because of the inherent uncertainty of outcomes and player seeding; indeed, if perfect ranking was already available at the outset, there would be little point to running a tournament in the first place. The author also provides discussion on the tournament outcomes and spectator interest; where players who play optimally are perceived to be dull or unimaginative.

5.9 "A new knockout tournament seeding method and its axiomatic justification" [9]

This paper demonstrates the determinativeness of seeding to single-elimination tournament outcomes, and proposes an "equal gap" seeding method contrary to the traditional "slaughter seeding" method, that, under a deterministic domain assumption, satisfies fairness, competitive integrity, and equal rank difference axioms that are introduced. This assumption is roughly summarised as that for any game m with players a and b, the player with the highest seed (skill) shall win, so the usefulness of equal gap seeding in practice is not completely clear to me. It will be a research aim of my project to simulate the effect of different seeding methods on tournament outcomes; equal-gap seeding provides a plausible alternative to the standard elimination seeding method.

5.10 "The efficacy of tournament designs" [10]

This paper provides a great template for how statistical analysis and simulation can be performed to demonstrate the superiority of particular tournament structures over another in terms of given metrics. In particular, Swiss-style tournaments are shown to be quite effective when compared to single or double elimination tournaments, using generated Elo distributions as well as real-world data from chess, soccer, and tennis.

Efficacy of tournaments is analysed in terms of ranking inversions exhibited at the end of the tournament, which matches my intuition that tournament structures can be expressed as sorting problems, and complements the aims of this project quite well. A valuable result is that triple-elimination does not greatly improve the efficacy of ranking players compared to double-elimination, especially when accounting for the extra matches required.

Swiss-style tournaments are shown to be very effective at ranking players and generally exhibit fewer inversions than any other format considered, for the same number of matches - although Swiss-style tournaments use a matching algorithm each round to determine who plays who, they are ran to a fixed number of rounds, so they can be engineered to desired level of accuracy and matches. Swiss-style tournaments are shown to be superior to single/double/triple elimination and group stage tournaments.

The choice of matching algorithm here likely has the greatest effect on result, which can be a result that my research project creates. Viewing Swiss-style tournaments as partially-evaluated sorting networks enables this view. Indeed, the comically titled sorting algorithm "I Can't Believe It Can Sort" [11], implemented as a mistaken version of insertion or bubble sort, can be viewed as a Swiss-style tournament with a matching algorithm that allows rematches, and ran to n rounds for n players; close "elements" - players - are repeatedly compared to create an accurate ranking of those players.

5.11 "Design Guidelines for Domain Specific Languages" [12]

This paper provides a list of guidelines to follow for the design of DSLs, that may be useful in this project as a DSL is a key artefact of it. In the case of this paper, it is not so useful here to "review" it in the sense that other literature is reviewed, but instead to respond to the guidelines posited, in order to validate the aims of the DSL that is to be designed.

The guidelines identified, and my response to each with respect to this research project's DSL, are:

1. **"Identify language uses early"**

   The use of the language is identified in the introduction section of this literature review; the design and implementation of novel tournament structures, that maximise various metrics (ranking precision, competitiveness, fairness, etc).

2. **"Ask questions"**

   - **"Who is going to model in the DSL?"** Myself, and tournament organisers who may find the software useful.
   - "Who is going to review the models?" Myself, and tournament organisers who may find the software useful.
   - "When?" During or near to project artefact completion, for use in creating analysis that will be reported on in the final thesis paper.
   - "Who is using the models for which purpose?"
     - For myself: To identify and analyse the efficacy of various :tournament structures
     - For other tournament organisers: Run novel tournament structures with real players

3. "Make your language consistent."

   The DSL should borrow existing semantics of existing tooling as much as possible. I believe that semantics similar to GraphViz DOT will be appropriate, with the addition of supporting mathematical operations, and some looping or recursion constructs.

4. "Decide carefully whether to use graphical or textual realization"

   A textual representation will be the primary format for this DSL, due to the extra effort required to design a visual system. However, it is noted that existing tournament structures are often fully "visual" in nature; a tournament organiser may elect to use a pen and paper to draw a tournament structure and the progression of players through it. Therefore, visualisation of the tournament structures after creation from the DSL may be provide some value.

5. "Compose existing languages where possible", and,

6. "Reuse existing language definitions" As above, the language design will take significant cues from GraphViz's DOT format.

7. "Reuse existing type systems" This guideline raises the question of whether implementation of an eDSL may be appropriate or not, as an eDSL can re-use the type system of its host language, which may be quite valuable if the type system is fairly expressive such as in a language like Haskell.

6 A Library for Tournament Design

6.1 Goals

1. Easy to use
2. Minimal number of primitives
3. Allow for analysis -> Statically analyse the pure subsets of a tournament
4. Be able to express common tournament formats

6.2 Designs that were explored

- Indexing of triangular numbers
  - A round-robin schedule has a triangular number of games. All rounds of all tournaments are some subset of a round-robin schedule. So you could probably express tournaments as an index into this domain; but how do you decide those indices? You probably end up needing or wanting an eDSL anyway to get those.
- Match lists
  - Just use `List (List Match)`. What tournaments can't be expresed like this?
- Steps + Step builders
- Limitations leading to final design

6.3 A type for Tournaments

Algebraic graphs primer / credit to the idea

6.4 Tournament primitives

- Type-encoded tournament depth
- Overlays
  - Meaning of at single round level
  - Meaning of at rounds level
- Sequences

    - Meaning of at rounds level

- Focuses
    - Meaning of at single round level
    - Meaning of at the rounds level

- Sort methods
    - Meaning of at single round level
    - Meaning of at rounds level

6.5 A monad for constructing a Tournament

List-monad for accumulating tournaments

- "Inverting" the Tournament type with continuations, avoiding lambdas
- Is it a law-abiding Monad?

6.6 Compilation into sorting network-like structures

6.7 Optionally-dependent streams of matches

We have streams of matches that can depend on an outside monad like IO to get match results, but they can also run completely purely. They can be inspected and ran without going through IO to say whether they are pure or not; the pure "prefix" of the tournament can then be returned, along with the impure "suffix".

6.8 An instruction set for tournaments

6.9 Why are some tournaments not sorting networks?

Normal vs point-award sorting networks. SE/DE are sorting networks, they have a static description using matches where the winner takes the high position in the sorting network. But round-robin schedules are also static descriptions, yet they are not sorting networks; any "upsets" in the round-robin schedule would cause it to perform a different tournament when expressed as a swapping sorting network. I.e. it needs the players to only exchange positions after all matches are complete. Since we are still able to inspect round robin tournaments statically, whether or not a tournament is a normal sorting network is not a good litmus test.

Call "pure" tournaments those which have a static description that do not depend on the standings of the tournament for all matches to be output. Call "impure" ones tournaments that require standings to know all the matches. Since we can add a primitive to change the sorting method, round-robin tournaments are "pure".

6.10 Simulators of tournaments

6.11 Pure simulator

6.12 Statistical simulator

6.13 IO simulator

6.13.1 Piped

6.13.2 Interactive

7 Tutorial

Each to have a short description, a visualisation of the generated sorting network, and the code required to describe them.

7.1 Specification of Common Formats

7.2 Single and Double Elimination

- “Slaughter” seeding

7.3 Round Robin

7.3.1 Round robin using standings

7.3.2 Sorting methods

7.4 Swiss-style

7.5 Common Compositions

7.6 Groups of N

7.7 Groups of N, then combined back again

7.8 Accept only the top N players of the previous sub-tournament

7.9 Sorts

7.10 Insertion sort

7.11 Bitonic sort

8 Analysis

8.1 Rank preservation

Assign elo to players and use that to simulate match results. At the end of the tournament, compare expected (players sorted by elo) to actual. Use normal elo distribution.

9 Availability of this library

Link to online documentation, package, source repository, etc.

10 References

[1] I. Stanton and V. V. Williams, “The structure, efficacy, and manipulation of double-elimination tournaments,” *Journal of quantitative analysis in sports*, vol. 0, no. 0, pp. 1–17, Jan. 2013, doi: 10.1515/jqas-2012-0055.

[2] C. T. Edwahttps://doi.org/10.1515/jqas-2012-0055rds, “Double-elimination tournaments: Counting and calculating,” *The american statistician*, vol. 50, no. 1, pp. 27–33, Feb. 1996, doi: 10.1080/00031305.1996.10473538.

[3] A. V. N. Dinh, N. P. H. Bao, M. N. A. Khalid, and H. Iida, “Simulating competitiveness and precision in a tournament structure: a reaper tournament system,” *International journal of information technology*, vol. 12, no. 1, pp. 1–18, Nov. 2019, doi: 10.1007/s41870-019-00397-5.

[4] N. Pham, H. Bao, S. Xiong, and H. Iida, “Reaper tournament system,” 2017, pp. 16–33. doi: 10.1007/978-3-319-73062-2_2.

[5] L. Csató, “Quantifying the unfairness of the 2018 fifa world cup qualification,” *International journal of sports science &amp; coaching*, vol. 18, no. 1, pp. 183–196, Apr. 2022, doi: 10.1177/17479541211073455.

[6] J. Guyon, “Risk of collusion: Will groups of 3 ruin the fifa world cup?,” *Journal of sports analytics*, vol. 6, no. 4, pp. 259–279, Jan. 2021, doi: 10.3233/jsa-200414.

[7] D. Van Bulck and D. Goossens, “Handling fairness issues in time-relaxed tournaments with availability constraints,” *Computers &amp; operations research*, vol. 115, p. 104856, Mar. 2020, doi: 10.1016/j.cor.2019.104856.

[8] P. C. Placek, “The impossibility of a perfect tournament,” *Entertainment computing*, vol. 45, p. 100540, Mar. 2023, doi: 10.1016/j.entcom.2022.100540.

[9] A. Karpov, “A new knockout tournament seeding method and its axiomatic justification,” *Operations research letters*, vol. 44, no. 6, pp. 706–711, Nov. 2016, doi: 10.1016/j.orl.2016.09.003.

[10] B. R. Sziklai, P. Biró, and L. Csató, “The efficacy of tournament designs,” *Computers &amp; operations research*, vol. 144, p. 105821, Aug. 2022, doi: 10.1016/j.cor.2022.105821.

[11] S. P. Y. Fung, “Is this the simplest (and most surprising) sorting algorithm ever?” arXiv, 2021. doi: 10.48550/ARXIV.2110.01111.

[12] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel, “Design guidelines for domain specific languages.” arXiv, 2014. doi: 10.48550/ARXIV.1409.2378.