바둑학연구, 제18권 제2호, pp. 162
Journal of Go Studies
2024. Vol.18, No.2. pp. 51-70

Pairgoth: A Modern and Flexible Software for Efficient Go Tournament Organization

Théo Barollet, Claude Brisson, Quentin Rendu

Abstract

Pairing players during a Go tournament is a complex task. One must register the

players, pair them using a pairing system, gather the results and display the information to the players. Several standalone programs offer these functionalities, however they are often game-specific and maintained and developed by a single person. The pairing itself is non-trivial, because of different pairing systems and of many parameters influencing them. This creates challenges for an intuitive user interface which can be used by non-experts tournament organizers.

In this article, Pairgoth is presented, a new pairing software inspired from Opengotha, a mainstream Go pairing software heavily used in Europe. Several improvements have been added to the pairing algorithm, which has also been made more generic. New pairing systems can easily be implemented in Pairgoth. Although initially designed for Go, it can easily be used for other games such as Chess, Shogi or Scrabble. Pairgoth consists of a pairing en-

gine coupled with a web-based user interface. This allows management of the tournament from several machines, including smartphones. It already supports Swiss and MacMahon pairing systems, while more options are currently under development (Round-Robin, accelerated Swiss, Amalfi, …).

Pairgoth was tested in real conditions at the international Grenoble tournament (TIGGRE 2024, 5 rounds Mac-Mahon tournament with top group and super top-group, 158 players from 29k to 7d) and at international Paris tournament (51st TIP, 6 rounds Mac-Mahon tournament with top-group, 160 players from 30k to 8d). Pairgoth was also successfully used during the 2024 European Go Congress in Toulouse, where nearly a thousand players participated. It was used for the prestigious main tournament as well as the majority of the side tournaments. It was recently used in the 2024 KPMC edition, making Pairgoth used in several countries.

On top of presenting Pairgoth, this article also tackles challenges encountered in pairing engines such as deterministic randomness, non-uniqueness of pairings, and the computation of fair standing criteria.

I. Introduction

To our knowledge there is no commercial tournament organizing program for the game of Go. These programs are often developed by individuals and shared gracefully with the Go community. To our mind, this brings two problems:

- We cannot expect a single individual to be an expert in developing user interfaces, manipulating the underlying graph theory for pairing players, writing a correct and bug-free pairing engine and having a lot of experience in tournament organizing so the software can solve all or nearly all the real case problems we can encounter in a tournament. These skills are summarized in Figure 1 and we believe it is quite improbable that they can be mastered by a single person.
- It is difficult to keep the time and motivation to develop a Go software in the long run so some tournament softwares are still in use today but cannot be maintained anymore.

To develop Pairgoth, we kept the volunteer work model but we involved from the ground up several people in the development process to tackle these two issues. We tried to have multiple people for each skill represented in Figure 1 so that someone can leave the project or have a break from it without the whole project being stopped.

Figure 1: Tasks to develop a pairing software, the items on this mind map will be discussed throughout this article.

Motivations and other pairing programs

The main motivation for the development of *Pairgoth* is the 2024 European Go Congress (this will now be referred to as EGC 2024). The name Pairgoth is a blend word between the word “pairing” and “*Opengotha*”: a free pairing software unfortunately not maintained anymore. This software was used in many tournaments in Europe and we use it as a basis for *Pairgoth.*

The currently used pairing programs are not designed to handle such a big event with more than a thousand players. We lacked at least two features to solve our real case problems:

- Several referees should be able to enter results remotely at the same time in the software, because we will have several hundred results per round.
- The software should be usable with a smartphone so we don’t need to give the private wifi access to everyone.

For example, Opengotha should be able to manage a thousand players but the tournament is accessible from a single machine so all the results must be entered on the same “master computer”. Its interface is not intuitive for new users but this would not have been a problem since we can rely on several expert tournament organizers at EGC 2024.

Another popular pairing program is MacMahon but it would have the same pitfall as Opengotha and has fewer tournament parameters.

An online software developed in North America by the website baduk.club caught our interest because it can be used online but the software is not mature enough and offers only a few tournament customization (https://baduk.club/tools).

II. Pairgoth architecture and challenges

The requirements for Go tournaments (including EGC 2024) are clearly understood, but it is not yet known if new use cases will arise. Additionally, Pairgoth is planned to be extended to other games that currently lack good pairing software, which may introduce unexpected use cases. Therefore, a modular architecture that can be easily extended in the future must be maintained.

We followed the primordial principle of the separation of concerns. Pairgoth comprises two distinct web applications:

1. an API Webapp, which encapsulates the pairing engine itself and exposes a REST API using JSON as in and out data formats.
2. A View Webapp, which exposes the HTML/Javascript web interface.

In a standard use case, both web applications are running inside the same web container, and custom automation tools (to import registered players, to publish results, to send notification emails, etc.) can easily be crafted:

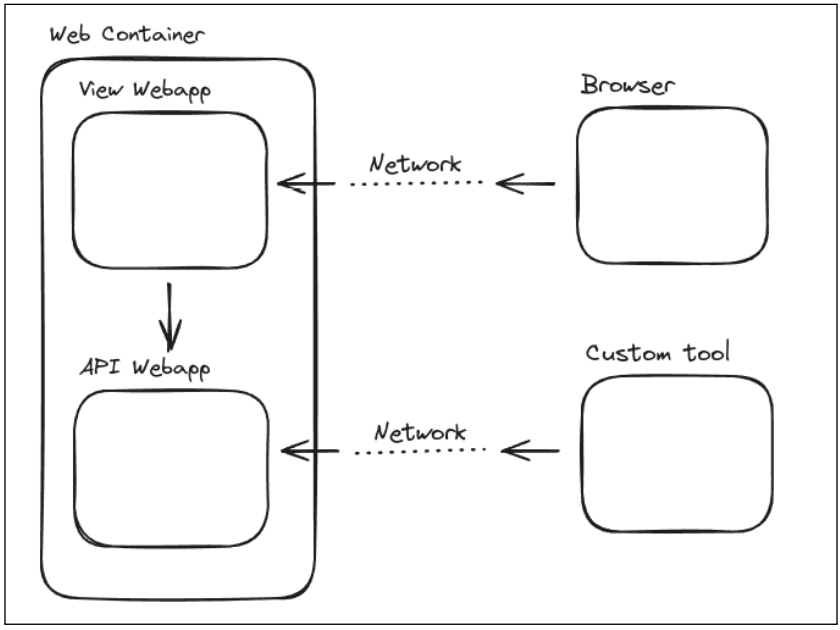

Figure 2a: Setup used for EGC2024, the left side is the server part and the right side can be multiple clients. The pairing engine is contained in the API Webapp.

Since the interface has been coded using a responsive layout, Pairgoth can already be used on a handheld device, but running inside a full-fledged mobile application is a future goal. For a mobile application, the browser and the View Webapp will both be embedded on the client side:

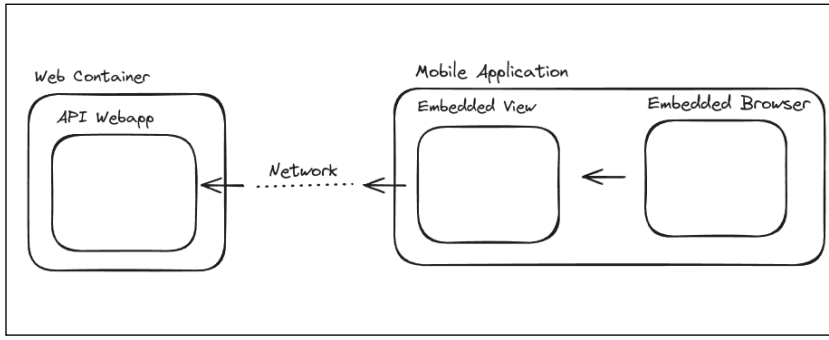

Figure 2b: Details of the mobile version of the application. The server is running on another machine (left side) and the mobile phone is represented on the right.

It's also worth noting that a web client/server application allows a de facto multi-user mode. In its current state, one user has to refresh its page to see changes committed by others. Future versions will make use of the Server Sent Events technology to have all changes be visible in real time.

The authentication layer is fully configurable and can be one of the following modes:

- "none" for when no authentication is needed, for instance, when running ocally
- "sesame" to share a single password among a tournament organizers, ideal when *Pairgoth* is running on the local network
- "oauth" to allow a single-sign-on process by the use of an external authentication server

We see in Figure 2a that the majority of the architecture doesn't depend on the game played like Chess, Go or Scrabble and only the pairing engine

contained in the Webapp can change. So if we want to extend Pairgoth with a rare pairing system for a specific case or a specific game we don't have a lot of work to do. Once all the software is stable, we plan to add several new pairing algorithms and we wish external developers will be able to add their own and share them, as well as any useful import/export or publication tool using the API.

III. The pairing engine

Given a list of players and a set of criteria, the task of the pairing engine is to output a list of games which best fits the criteria. These criteria take into account the pairing system used (Swiss, Mac-Mahon, Round-Robin, …) as well as tournament parameters (handicap correction, avoiding intra-club pairings, …). In this section, a detailed description of the pairing algorithm is given.

1. Data representation

Following *OpenGotha* and *Mac-Mahon* softwares, a graph structure is used in Pairgoth. A graph is a mathematical object consisting of points (called vertices) linked together by lines (called edges). Each player is associated with a vertex and a game between two players is represented by the edge linking their vertices. An example of the graph representation of a pairing is shown in Figure 3(a). The graph represents 5 players (A to E) between which 2 games are played (A vs. C and B vs. D). Because of the odd numbers of players, player E is not paired (E is the bye player). Given the same 5

players, many other pairings are possible. In the next subsection, the general algorithm to find the best pairing is presented.

2. General pairing algorithm

To find the best pairing, weights are associated with each possible game. Games fitting the tournament criteria will be associated with a large weight, whereas unwanted games will be given a small weight.

The first step of the algorithm is to compute the weights for all the possible games. For each player *i*, one needs to loop over all the other players j and to compute w(*i,j*) the weight associated to a game between i and *j*. Notice that for N players, the weight function is called *N*(N-1)* times. A description of the weight function is given in the next subsection. A simplified example is shown in Fig. 3b where weights have been assigned to all the edges. In this simplified example, the weight has been assumed symmetric, whereas it might not be the case. If one of the criteria is to balance the number of times each player gets to play Black and White, *w(i,j)* will be different from *w(j,i)*.

The second step of the algorithm is to find a pairing involving all the players which maximizes the sum of weights. In graph theory, this is called a maximum weight perfect matching. In *Pairgoth*, this optimization step is performed by the external library *jgrapht* (Dimitrios2020) relying on the Blossom V algorithm (Kolmogorov2009). The algorithm is guaranteed to give a maximum weight perfect matching. However, the solution might not be unique. If several maximum weight perfect matchings coexist, only one is returned.

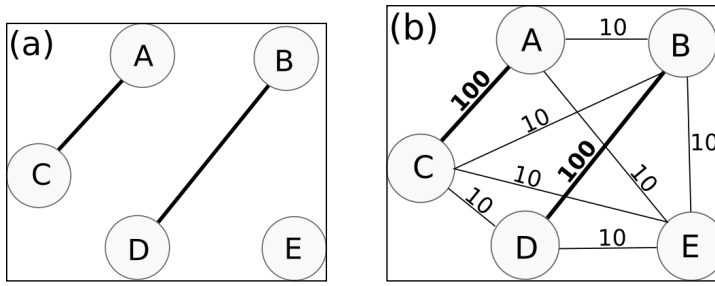

Figure 3: (a) A graph representation of pairings involving 5 players and two games (A vs. C and B vs. D) (b) A graph representation of all the possible games with the associated weights. The maximum weight perfect matching is shown in bold (A vs. C and B vs. D). E is the bye player.

3. Weighting function

Most of the complexity of the pairing engine lies in the weighting function. Pairgoth's weighting function is designed to reproduce OpenGotha evaluation function. This was especially useful during testing and debugging.

The weight associated with a game is computed as a sum of weights associated with different criteria. The exhaustive list of the criteria, ranked from most important to last, is the following (along with the associated function in Pairgoth 0.14):

1. players did not already play each other (avoidDuplicatingGames)
2. minimize score difference (minimizeScoreDifference)
3. add randomness to the pairings (applyRandom)
4. balance drawn-up and drawn-down (applyDUDD)
5. apply Split & Slip/Fold/Random (applySeeding)

6. balance White and Black for each player (applyColorBalance)
7. avoid intra-club/country pairings (Geographical.apply)

To tackle the importance of the different criteria, each of them has a defined maximum weight. The largest maximum weights are associated with the most important criteria. For instance, for a typical Go Mac-Mahon tournament, the maximum weight associated with avoiding duplicate games is 5000 times larger than the weight minimizing score difference, which is itself 100 times larger than the weight applying randomness. It is in general not recommended changing these maximal values, but it might be necessary when adapting Pairgoth to other games, or to modify the relative importance of the criteria.

In the next paragraphs, all the criteria are presented, as well as their main parameters. They are described in the default order presented above.

The first criterion makes sure that two players do not play more than one game in a given tournament. This can be deactivated, for instance, when organizing a Double Round-Robin tournament.

The second criterion minimizes the score difference between two paired players. In a Swiss system, it makes sure that players with the same number of wins will play together. In a Mac-Mahon tournament, it minimizes the Mac-Mahon score difference between two paired players, hence naturally minimizing the handicap (if any).

The third criterion allows for randomness. If selected, the players with the same score will be paired randomly and not along a standard Split & Slip/Fold/Random. If deterministic randomness is chosen, a given set of tournament parameters and players will always give the same pairings. Otherwise,

the pairings will be different.

The fourth criterion, referred to as Draw-up/Draw-down, handles odd groups of players. In that case, a player, called a floater, must be removed from the group and added to another group. A detailed description of this criterion and its parameters is given in the next subsection.

The fifth criterion applies a Split & Slip/Fold/Random (referred to as a seeding) inside a group of players with the same score. The three different seedings are described in Figure 4. Two different seedings can be used during a tournament, for instance Split & Random for the first two rounds and Split & Split for the remaining rounds.

The sixth criterion balances the number of times a player is playing Black or White.

The seventh criterion can be applied to avoid pairings between players of the same club or the same country. It can be parametrized using the preferred score gap. For instance, an intra-club pairing preferred score gap of 3 (default value) means that the pairing engine will prefer a game between players of different score (up to a difference of 3) over a game between players of the same club. Additional parameters include the threshold above which this criterion is not applied (for instance in the top group of a Mac-Mahon tournament).

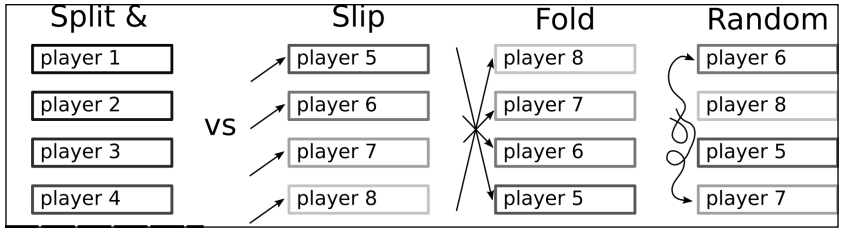

Figure 4: Sketch representing the three types of seeding. A group of ordered players (8 here) with the same score is first split into two equal parts who will play against each other. Keeping the initial order for the first part, the order of the players in the bottom part defines the type of seeding. In Slip, the order is unchanged. In Fold the order is reversed. In Random the order is randomly assigned.

Most of the pairing parameters are accessible through the UI. On the information page of the tournament are found the most important parameters, such as the system (Swiss, Mac-Mahon, ...), the number of rounds or the handicap settings. In Edit mode, the advanced parameters appear at the bottom, allowing users to tweak the seven criteria mentioned above. A sketch is shown in Figure 5.

Figure 5: Sketch showing where to find all the pairing parameters in *Pairgoth's* UI

4. Draw-up draw-down behavior

The draw-up/draw-down (DUDD) criterion might lead to non-intuitive behavior. This criterion is needed to deal with floaters, i.e. extra players in odd groups of players. One set of parameters deals with the choice of the extra players. One can choose to remove the player at the top, in the middle, or at the bottom of the group. This can be done independently for the upper and the lower groups. These choices are mandatory and are likely to affect the results of a tournament. In *Pairgoth*, the middle/middle choice is the default value.

In any case, a player which has previously been drawn-up (or drawn-down) is unlikely to be drawn again in the same sense. By default, *Pairgoth* will try to compensate a previous draw-up by a draw-down, and vice versa. This compensation can be deactivated in the advanced parameters.

When choosing to compensate for previous DUDD, it is recommended to

have the same parameter for the choice of the floater in both upper and lower groups (for instance middle/middle). In a Mac-Mahon tournament with a top group, a bottom/top choice can be made to protect the SOSOS of drawn-down top group players. However, if combined with the compensation for previous DUDD, the drawn-up players from the top of the second group will be drawn-down in the next rounds.

5. Towards fair standings in Go tournaments

When organizing a tournament, a compromise should be found between offering an enjoyable experience to all the players and having final standings which best reflect players performance. Some players might not be able, or might not choose, to play all the rounds. The pairing system should be robust enough to allow this without penalizing the players whose opponents did not play all the rounds. Notice that for some important events, playing all the rounds is a criterion to enter the top group. When it is not the case, or for the other players, one needs to define fair standings criteria. Without loss of generality, a Mac-Mahon tournament will be discussed in the next paragraph, as it is the most probable system for big tournaments.

The first standing criterion is the Mac-Mahon score (MMS), equal to the initial score plus the number of wins. A player skipping a round is awarded half a point of MMS (this can be adjusted to 0 or 1 in the advanced parameters). The MMS is not affected by opponents skipping a round.

A common second criterion is the Sum of Opponents Score (SOS), which is the sum of opponents MMS. A player skipping a round does not have an opponent for the said round, which will lower its SOS. To overcome this

issue, its initial MMS is added to its SOS for each skipped round. This is the default behavior, as well as OpenGotha's behavior. In Pairgoth, a new available option is to add its initial Mac-Mahon score plus half the number of rounds (corresponding to a 50% chance of winning each round).This method is believed to be fairer, especially in tournaments with a large number of rounds such as the European Go Congress Main Open tournament.

A common third criterion is the Sum of Opponents SOS (SOSOS). Thanks to the correction applied to the SOS of players who skipped rounds, the SOSOS can be computed naturally.

IV. Results and discussions

1. From beta to a robust software

Achieving a bug-free implementation of *Pairgoth* for EGC 2024 necessitates extensive testing across various tournaments of different sizes. We have shared *Pairgoth* with all tournament organizers in France and established a mailing list (pairgoth-dev@jeudego.org) to gather feedback. Pairgoth is accessible at the following URL: https://pairgoth.jeudego.org/en/index-ffg. This initiative has allowed us to resolve many bugs within the Pairgoth pairing system, as well as some pre-existing issues in OpenGotha.

Before using *Pairgoth* on large scale international tournaments, it has been thoroughly tested. Its first successful event was the international Grenoble tournament (TIGGRE 2024), which hosted 158 players ranging from 29 kyu to 7 dan. At the Paris international tournament, although *Pairgoth* could not be used alone due to printing issues, the pairings were successfully verified

with those of OpenGotha. Several small to medium-sized tournaments have also used Pairgoth, and any bugs encountered in the pairing system were not critical, with tournament organizers occasionally able to manually correct pairings. These issues, which have now been solved, helped to build up *Pairgoth's* reliability.

2. Pairgoth's achievements at major international Go tournaments

Pairgoth was successfully utilized during European Go Congress 2024, held in Toulouse from July 27th to August 11th. It handled three large-scale Mac-Mahon tournaments: the Open (10 rounds, 869 participants), the Rapid (8 rounds, 465 players) and the Weekend (5 rounds, 619 participants). The software's web-based interface facilitated collaboration among tournament organizers, allowing simultaneous real-time updates of the results. This was particularly useful for the Weekend tournaments which involved several rounds per day. On-site feedback from the organizers allowed real time adjustments of the software, improving its robustness and flexibility.

Pairgoth was adopted for the pairings of the 19th Korea Prime Minister Cup (KPMC), held in Taebaek from 20th to 26th September 2024. This annual tournament, organized by the Korean Baduk Association, is a world-class event welcoming players from all around the globe. This year, 60 countries were represented. The players (from 6k to 7d) were competing during a 7 rounds Swiss tournament. The integration of Pairgoth marked a significant advancement in the tournament's pairing process. Traditionally, initial round pairings were determined by random draws, which occasionally led to unbalanced matchups. In order to give everyone a chance while ensuring a reasonable rank difference for each game, the Split & Slip system was chosen. It

resulted in more balanced pairings and positive feedback from participants. The software's web-based interface also enhanced accessibility and ease of use for the organizers.

Overall, feedback from tournament organizers indicates that the new user interface is more intuitive and less cumbersome. The smartphone version of Pairgoth was partly used during KPMC, but has not yet been tested on a larger scale.

Although Pairgoth has mainly been used by French tournament organizers, it is beginning to expand worldwide, with notable adoption in Germany and the USA. Apart from French and English , the software is currently available in German and in Korean. We hope to see increased usage in the future, making it a new standard of Go tournaments organization.

V Conclusions

Pairgoth, a flexible pairing software, has been presented in this paper. It is currently used for efficient organization of Go tournaments, but can also be used for other games such as Chess, Shogi or Scrabble. Pairgoth is free and available under an open source license at https://pairgoth.jeudego.org/en/index-ffg. Pairgoth relies on a modular architecture, allowing easy maintenance and the possibility to add new pairing systems with minimal development. Features belonging to other games can also be easily implemented.

Pairgoth has been successfully used in several tournaments organized in France (including Grenoble and Paris international tournaments). It already reached users in Germany and the USA, and might be used for the Korean Prime Minister Cup (KPMC), showing a growing international community of

users.

Pairgoth stands out due to its innovative features, such as managing a tournament through multiple clients, the solidity of its theoretical basis, and the adaptability of its architecture. While Pairgoth is already a success, this achievement represents only the first step in the development of a modern and flexible pairing software that we hope will attract a large, international community of users, maintainers, and developers.

VI. References

Michail, D., Kinable, J., Naveh, B., & Sichi, J. V. (2020). JGraphT—A Java library for graph data structures and algorithms. *ACM Transactions on Mathematical Software (TOMS), 46*(2), 1-29.

Kolmogorov, V. (2009). Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation, 1*, 43-67.

Received: 9, November, 2024
Accepted: 18, November, 2024