TAMPERE UNIVERSITY OF TECHNOLOGY
Faculty of Computing and Electrical Engineering

TAPIO VUORINEN

Enhancing Go tournament pairings in Europe

Master thesis

Topic accepted in the faculty council of the Faculty of Computing and Electrical Engineering 02/03/2010

Examiners: D.Sc. Henri Hansen
Professor Tapio Elomaa

Foreword

This master thesis, *Enhancing Go tournament pairings in Europe*, has been written on my free time out of pure interest on the subject.

Over the last decade, Go has become more and more popular all over the world. This can be seen in larger tournaments and especially more players playing in tournaments abroad. During the last few years Go tournaments in Finland have evolved from ten or twenty players to a hundred. Larger tournaments need a lot more work. This is also the case in pairing, which is the process where the players are assigned their opponents.

This thesis is not a scientifically exact research to sort tournament managing programs to an order of any kind. It is just an attempt to enlighten the big and small differences between different kinds of solutions and to introduce a few new ideas about how to make the pairings give out a bit more information than we are normally used to.

I'd like to thank Teemu and Suvi Rovio, Olli Lounela, Tuomo Salo, Geoff Kaniuk, Anniina Salo, Henri Hansen and many more for giving me a lot of fine ideas, proof-reading and pushing me on.

Tampere, May 23rd 2010

Tapio Vuorinen
Hämeenpuisto 17-19 A 1
33210 TAMPERE
FINLAND

Contents

TAMPERE UNIVERSITY OF TECHNOLOGY
Faculty of Computing and Electrical Engineering
Institute of Software Systems

VUORINEN, TAPIO: Enhancing Go tournament pairings in Europe
Master of Science Thesis, 47 pages, 4 enclosure pages
Examiners: Professor Tapio Elomaa and D.Sc. Henri Hansen
May 2010
Keywords: Go, McMahon, Swiss, tournament, pairing

Go's popularity has growed constantly in both Finland and the whole Europe. Several pieces of software have been developed to pair and manage the tournaments and they differ in implementation, used algorithms and goals.

Algorithms based on weighted graphs where nodes represent the players and the edges represent possible pairs, have gained ground since Christoph Gerlach released his *MacMahon* software, but this field still has a lot of research left mostly in tuning the edge weights to obtain more information. The edge weights tell the quality factor of the possible pairs.

New algorithms or solutions to pairing Go tournaments have not been introduced for a few years and currently the tournament management programs tend to concentrate in tournament management and pairing is just a small part of a large feature set.

This thesis takes a look at the most common problems of pairing and presents a few solutions to them.

TAMPEREEN TEKNILLINEN YLIOPISTO

Tieto- ja sähkötekniikan tiedekunta

Ohjelmistotekniikan laitos

VUORINEN, TAPIO: Enhancing Go tournament pairings in Europe

Diplomityö, 47 sivua, 4 liitesivua

Tarkastajat: Professori Tapio Elomaa ja yliassistentti Henri Hansen

Toukokuu 2010

Avainsanat: go, turnaus, paritus, McMahon, sveitsiläinen

Go on kasvattanut jatkuvasti suosiotaan sekä Suomessa että koko Euroopassa. Turnausten hallintaan ja pelien määräämiseen on kehitetty useita eri ohjelmistoja, jotka eroavat hieman toteutustavoiltaan, käyttämiensä algoritmien myötä ja tavoitteiltaan.

Parien painotukseen perustuvat graafialgoritmit, joissa solmut edustavat pelaajia ja kaaret mahdollisia pareja, ovat vallanneet merkittävästi alaa Christoph Gerlachin julkaistua *MacMahon*-paritusohjelmansa, mutta tälläkin saralla on vielä runsaasti kehitettävää lähinnä kaarien painotuskertoimien säädössä informaation lisäämiseksi. Kaarien painoarvot ovat mahdollisten parien hyvyyslukuja.

Turnausten paritukseen ei ole esitelty uusia algoritmeja tai näkökulmia muutamiin vuosiin ja uudet turnausohjelmistot keskittyvätkin pääasiassa turnausten hallinnointiin ja paritus on vain yksi osuus pitkästä ominaisuuslistasta.

Tässä työssä kartoitetaan yleisimmät parituksen ongelmat sekä esitellään ja pohditaan erilaisia ratkaisuja näihin.

Abbreviations, terms and definitions

AGA Amerigan Go Association, http://www.usgo.org/

AI Artificial Intelligence

board In addition to being an actual part of the equipment in Go, a board is a tournament term. Boards are numbered from 1 to $N/2$ where N is the number of participants. The boards are ordered so that the leading player (and of course his opponent) gets to to play on board one, the next player from the top yet without a board on board two and so on.

deflation In economics, decrease of the money supply relative to the amount of goods and services. In Go, the increase of players' strength relative to their rating or grade.

EGF European Go Federation, http://www.eurogofed.org/.

GoR Go rating based on tournament results, calculated by EGF.

grade Describes the player's strength with a scale from 30 kyu to 7 dan. Professional grades scale from 1 dan to 9 dan.

minimax tree search A form of game AI. A search tree is formed by playing out all the possible move combinations and the one leading to the best result for the player and the worst to the opponent is chosen. Mimimax consumes a lot of memory and is not very fast.

pairing A pairing is a data set telling who is playing who and the players who are absent or on a free round. Each round of a tournament has a pairing.

rating An estimate of player's strength calculated by some pre-determined formula, e.g. GoR.

POSIX "Portable Operating System Interface" is the collective name of a family of related standards specified by the IEEE to define the application programming interface (API) for software compatible with variants of the Unix operating system, although the standard can apply to any operating system. [9]

Chapter 1

Introduction

1.1 The goal of this thesis

The goal of this thesis is firstly to study problems encountered in pairing a Go tournament using the McMahon system. Secondly, to introduce the reader to the solutions suggested so far by looking at the most often used pairing programs and some new ideas on how to improve pairings in all of Europe.

1.2 The structure of this thesis

First, the reader is familiarized with the game of Go itself and grading in Chapter 2. Chapter 3 presents the most common tournament systems. The main theory of this thesis is in Chapter 4, which focuses on McMahon system. Chapters 5 and 6 present a couple of solutions to the problems stated earlier by taking a look at some tournament management programs. Chapter 7 includes test runs made with the software listed in Chapter 5 and, finally, Chapter 8 gives the conclusions.

1.3 Go

Originating from China, Go is an over 3000 years old boardgame. It is a deterministic, perfect information game of strategy between two players. The name “Go” comes from the

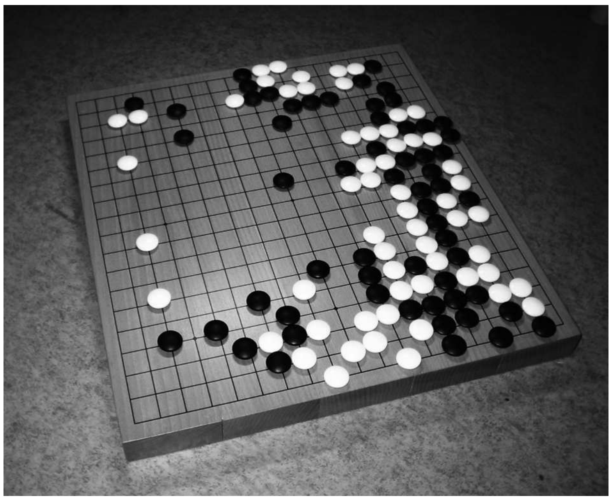


Figure 1.1: Go equipment

Japanese name “Igo”, which is roughly translated as “surrounding boardgame”. In China, the game is known as *wei-qi* and in Korea as *baduk*.

1.3.1 The rules of Go

The game is played with a board with 19×19 crossings and black and white stones. Smaller boards (13×13 or 9×9) can be used for teaching purposes. Black plays first. In a *handicap game* the black player is weaker than white and gets to play a certain number of stones before white’s first turn The positioning of the handicap stones is considered as black’s first move. [7]

Both players place one stone on a free crossing on the board during their turn. Players may also pass on their turn. If a continuous chain of stones connected along the lines on the board is surrounded by opponent’s stones so that there are no free crossings around it, the chain is removed from the board and the stones are counted as prisoners for the opponent. The main objective of the game is to surround more territory from the board than the opponent. Territory is defined as an area surrounded by the player so that his opponent can not play any stones inside it without them being eventually captured.

If both players pass in succession to inform they do not see any move left that would change the situation of the game, the game ends. Dead stones (stones that would eventually get captured anyway) are removed from the board and counted as prisoners. Both players get one point per each free crossing in their territory and one point for each prisoner they got. In even games white gets some points to balance the black's advantage playing the first move, nowadays 6.5 points. This is called *komi*. The player with more points wins. This type of point counting is used in Japanese rules used in Finland and most European countries, other rulesets may differ slightly but the basic principles are the same. [7]

1.3.2 Real life applications and computer Go

A strong analogy between Go and for example war has been drawn often, partly because of both strategic and tactic nature of the game. Small scale tactics in local situations may affect the whole board in a strategic sense. [10]

This dual character of the game is one of the reasons why no computer software able to beat even the best amateur players has been developed yet, evaluating the whole board situations is a quite difficult task even all human players cannot do. Computer programs usually use different methods of playing than traditional AI techniques such as *minimax tree search*. Recently for example neural networks and genetic algorithms as forms of machine learning have been taken into use. [12]

In December 2008, a program called Crazy Stone beat a professional player with the smallest handicap ever, seven stones [2]. Crazy Stone uses a Monte-Carlo AI algorithm [3]. In Monte-Carlo a list of potential moves is generated. For each move a large number (i.e. thousands) of games at random on the resulting board. The random games are evaluated and the move providing the best set of random games is chosen as the best move.

Chapter 2

Grading and tournaments

2.1 The grading system in Go

Go, like many martial arts, has a *grading system* involving *kyu* and *dan* grades meaning “class” and “grade”. A beginner having just learned the rules usually gets the grade 30 kyu. The grade then progresses up to 1 kyu following the player’s skills. Up from 1 kyu are 1 dan, 2 dan etc.

The grades are used for two purposes. First of all, in handicap games, the difference between the grades tells the amount of handicap stones needed. A 5 kyu player gives four handicap stones to a 9 kyu player, a 2 dan gives three stones to a 2 kyu and so on. Hence a 5 kyu player is said to be "four stones stronger" than a 9 kyu one. Second, some tournament systems have to be initialized with data based on grades.

The following is the minimal set of features required from a working grading system ordered by importance, highest first

1. Grades provide estimates of the winning probabilities for both players.
2. Grades can be tested with even (non-handicap) games with big grade differences. The stronger player should win most of the games.
3. Grades can be tested with handicap games. (Both players should win roughly an equal number of games.)

4. A stronger player's grade is more stable which means that his strength varies less than a weaker one's and therefore the game results are more predictable.

5. Grades drop very rarely and even then only slightly.

These features can and hopefully will be taken into account when designing grading (kyus and dans) or *rating* (numeric strength estimator) systems.

2.2 GoR

The Czech Go Association adopted the base system behind *GoR* to use for their own players and tournaments in early 1998. Later it was expanded by the Czechs to include other European tournaments and players. Since November 1998 the system has been used to compute the official EGF ratings. The rating list includes all the European players (and some from other continents) who participated in the tournaments in the database.

The ratings are matched to grades with 100 point intervals. The lowest possible rating is 100 points, which equals 20 kyu. If the rating drops below 100 it is reset back to 100. Average 1 kyu and 3 dan should have ratings of 2000 and 2300 points. When a new player not yet found in the database enters a tourmament, the rating is reset according to his grade. Professional ratings are reset to 1p (a pro 1 dan) = 2700, 2p = 2730 and so on with intervals of 30 points per grade.

Because of different kinds of grading systems across countries ratings do not correlate with GoR very well especially for lower kyu grades. However, in a closed system, the GoR system provides quite good estimates of the players' relative strength given that the database includes enough games for each player.

2.2.1 System description

Derived from *ELO* rating system used by International Chess Federation (FIDE), the GoR system is based on defining a probability of winning a game (*winning expectancy, S*) between players A and B for both players depending on the rating difference $D = gor_B - gor_A$.

This is only an estimate based on statistical information available. Let us call the player with lower rating *A*. His winning expectancy S_A is obtained from equation 2.1

$$S_A = \frac{1}{e^{\frac{D}{a}} + 1} \tag{2.1}$$

, where D is the rating difference, e the Euler's number and a a variable which depends on the player's rating. a is defined by $a(R) = 200 - \frac{R-100}{20}$ when $100 \leq R \leq 2700$ and $a = 70$ for $R > 2700$.

The winning expectancy of *B*, A's higher (or evenly) ranked opponent, is obtained from the equation $S_B = 1 - S_A - \epsilon$.

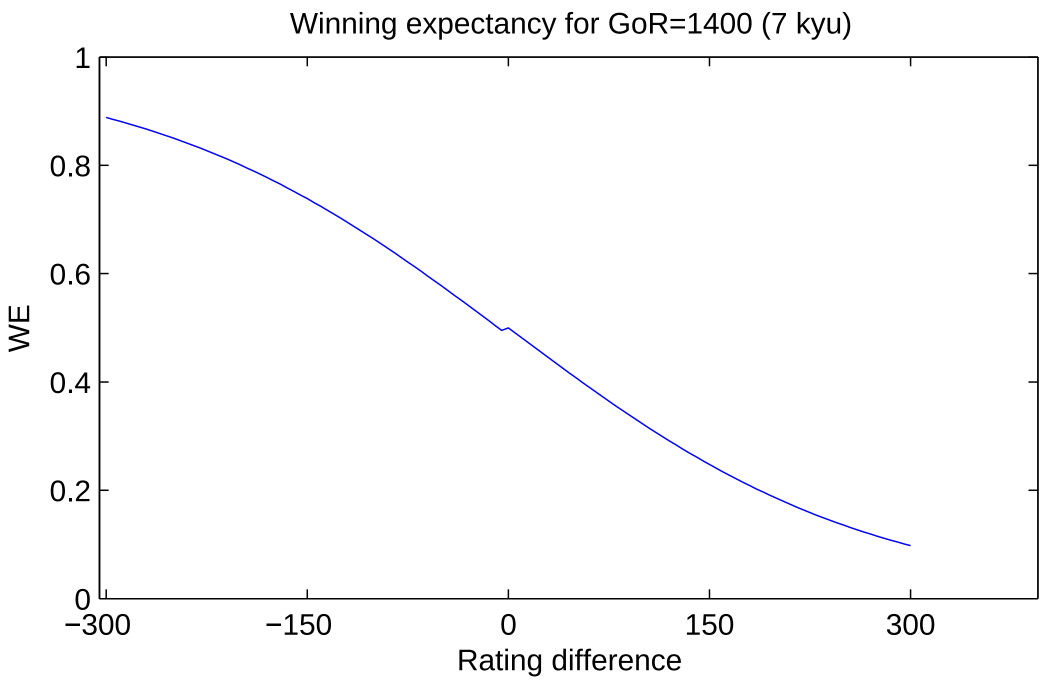
ϵ is used to counter the *deflation* (increase of players' strength relative to their rating or grade) effect introduced by new improving players taking points from established players. As the GoR system is used alongside with grades as compared to a stand-alone system for computing relative strengths, this is necessary so that the already established players can "keep up" with their grade level. Currently EGF uses $\epsilon = 0.016$, which was chosen to balance rating variations in dan region (GoR over 2100). Because the system has been adjusted to work optimally on dan players, the effect it has on the kyu players is way too small because the weaker players develop faster. This leads to deflation especially in countries whose players do not regularly play abroad. As the players become stronger the deflation is moved upwards.

See Figure 2.1 for winning expectancies for players with GoR 1400 and 2200. On the horizontal axis is the rating difference with the player in question (having GoR 1400 or 2200) and on the vertical axis the winning probability of the player. Handicaps are taken into account by adding 100 points for each handicap stone to the weaker player's GoR when calculating the winning expectancies.

The rating of a player changes in a single game by

$$\Delta R = R_{new} - R_{old} = con(R_{old}) \cdot (r - S) \tag{2.2}$$

where r is the result (1 for a win, 0 for a loss or 0.5 for a tie, *jigo*) and the strength related factor $con(R)$ defines the maximum change. con is a decreasing function of the player's rating which implements the greater stability of higher grades in the system (see statement four on page 4). For the values of con for different ratings, see table 2.1.



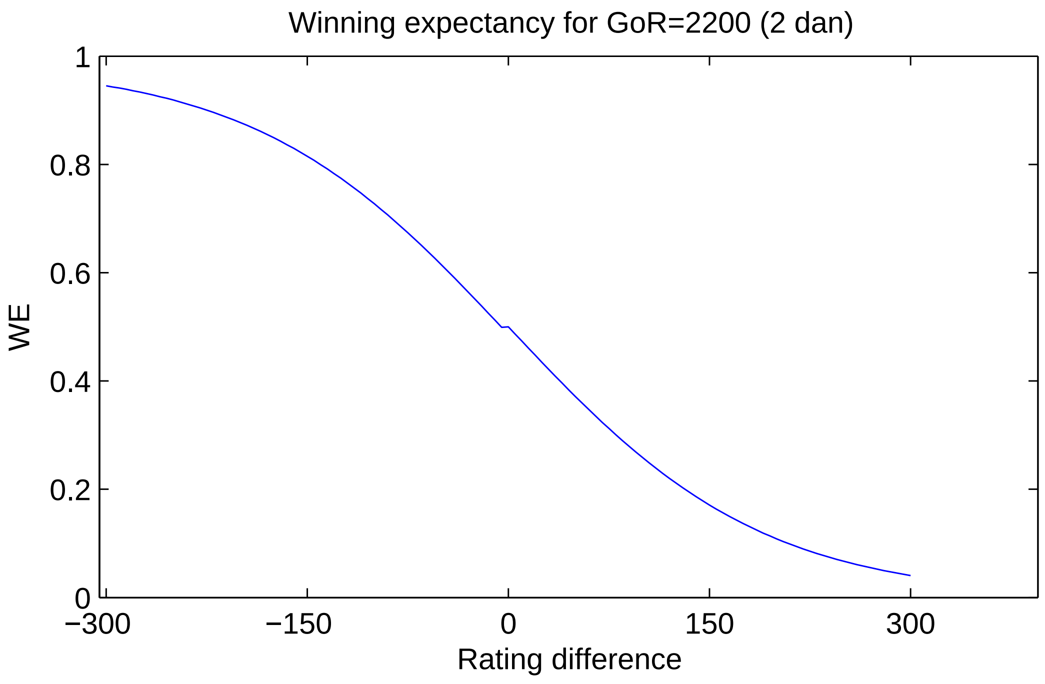


Figure 2.1: Winning expectancy (S) for players with GoR 1400 and 2200 as a function of rating difference D

2.2.2 Obtaining more rating information

To obtain more grading information in Finland, a player with more wins than losses is often paired with an opponent who is two to three stones stronger than the strongest opponent the player has already won. A similar method described below can be used also in countries which rely on GoR when deciding grades.

Ratings provide us with information about the players' relative strengths, "*A* is stronger than *B* but weaker than *C*". When looking at individual games, the changes in the rating can be thought of as correcting the ratings, i.e., moving them to the direction of the player's real strength. Because the mathematics behind GoR are known, we can calculate the strength difference which maximizes the total change of GoR in one game.

In the GoR system, using the equation 2.2, the expected value for the change of the weaker player's rating is

$E(\Delta GoR_A) = con_A \cdot (S_A(1 - S_A) + S_B(0 - S_A))$, where con_A is a value dependent on the player's rating (see the table 2.1), S_A is the player's winning expectancy, S_B is the winning expectancy of the stronger player. S_B is defined by

$S_B = 1 - S_A - \epsilon$

The expected value for the change of the stronger player's rating is

$E(\Delta GoR_B) = con_B \cdot (S_B(1 - S_B) + S_A(0 - S_B))$.

The expected value for the total change of GoR points is the sum of the expected values for players *A* and *B*:

$E(\Delta GoR_{total}) = con_A \cdot (S_A(1-S_A)+S_B(0-S_A))+con_B \cdot (S_B(1-S_B)+S_A(0-S_B))$

which can be simplifed:

Table 2.1: con for different ratings

Rating	**con**	**Rating**	**con**
100	116	1500	47
200	110	1600	43
300	105	1700	39
400	100	1800	35
500	95	1900	31
600	90	2000	27
700	85	2100	24
800	80	2200	21
900	75	2300	18
1000	70	2400	15
1100	65	2500	13
1200	60	2600	11
1300	55	2700	10
1400	51		

$$\begin{aligned}
E(\Delta GoR_{total}) &= con_A \cdot (S_A(1-S_A) + S_B(0-S_A)) \\
&\quad + con_B \cdot (S_B(1-S_B) + S_A(0-S_B)) \\
&= con_A(S_A - {S_A}^2 - S_A S_B) + con_B(S_B - {S_B}^2 - S_B S_A) \\
&= con_A(S_A - {S_A}^2 - S_A(1 - S_A - \epsilon)) \\
&\quad + con_B(S_B - {S_B}^2 - S_B(1 - S_B - \epsilon)) \\
&= con_A(S_A - {S_A}^2 - S_A + {S_A}^2 + S_A\epsilon) \\
&\quad + con_B(S_B - {S_B}^2 - S_B + {S_B}^2 + S_B\epsilon) \\
&= con_A \cdot S_A \cdot \epsilon + con_B \cdot S_B \cdot \epsilon
\end{aligned}$$

For a fixed rating, E can be plotted as a function of rating difference x followingly: Fix the rating of player A R_A. Calculate the rating of player B by $R_B = R_A + x$ and E with the ratings R_A and R_B. The biggest changes in GoR points occur when the rating difference is between 200 and 300 points as shown in Figure 2.2. The maxima have been indicated with a grey line. For a higher rating, the maximum is closer to zero difference due to the higher confidence in higher ratings. This is not a problem for the reason that the rating information is valued more in lower rating region because of the faster development of players.

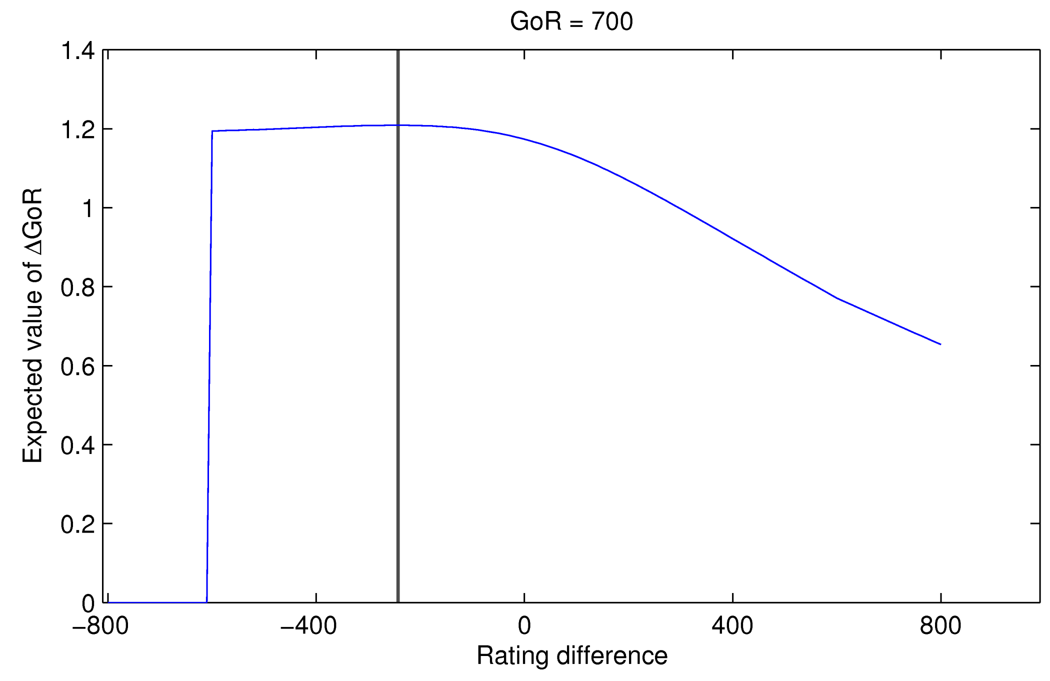
After two to three rounds, it is usually quite easy to see if a player's grade and rating are off by a stone or two, based on winning expectancies, i.e., the player has significantly more wins than losses. By pairing such a player with a player with a rating difference of 200-300 points (up or down, depending on the direction the player's rating needs to be corrected to), most information for the rating system (the largest change in ratings) can be obtained. Pairing two players with almost the same ratings means that 1) less information (smaller changes) is obtained and 2) if either of the ratings is greatly (100-200 points) off, the correction is too small. The optimal rating difference is smaller for higher ratings.

If a player enters a tournament with a too low a grade and thus has also a too low rating, his grade is usually off by one to three stones. A difference of less than one stone is unnoticeable and it is quite difficult to improve over three stones clandestinely. The optimality of the 200–300 point difference corresponds quite well with the method to obtain more grading information by pairing the players with too low grades to players whose grade is two to three stones higher than their strongest defeated opponent.

2.3 Grading systems in Europe

2.3.1 The grading system in Finland

In Finland players cannot decide their own grades above 10 kyu. Instead, they have to be promoted by stronger players. Promotion right are limited. These limits are bound to grades and are set by the Finnish Go federation. Because grading is based totally on human decisions, and no mathematical model is used, tournaments are used to obtain "grading information" on players with inaccurate grades. Grades give the expected winning probabilities for each game (see 2.1). If the result of the game is highly improbable – for example a player winning an opponent more than two stones stronger – the more probable it is that his grade is wrong.



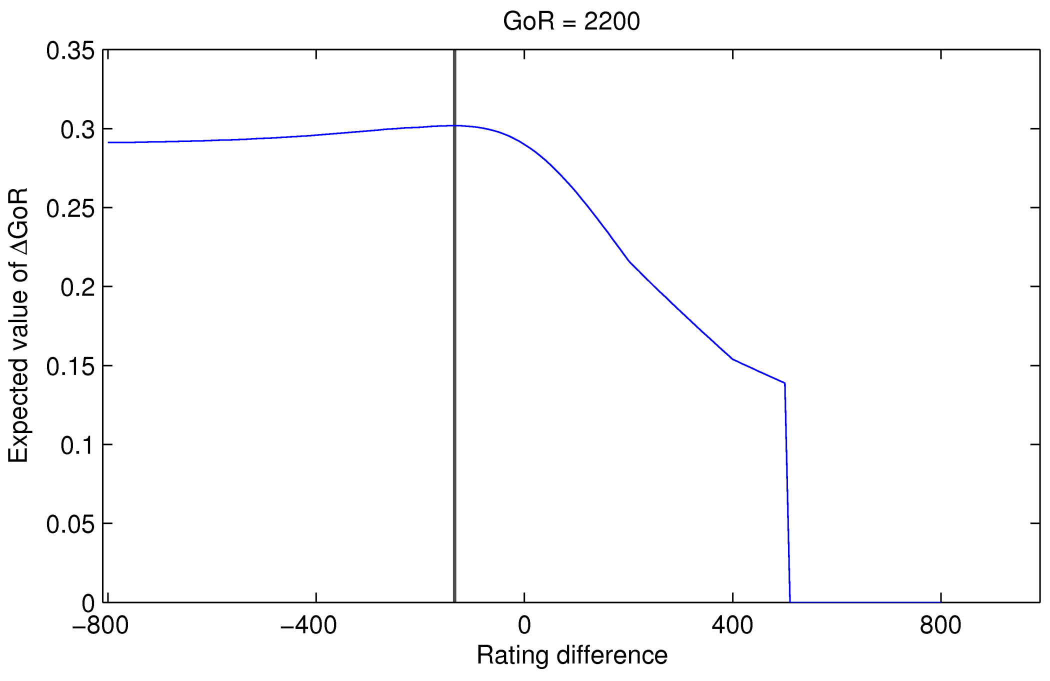


Figure 2.2: Expected value of total GoR change as a function of the rating difference with player's GoR either 700 or 2200.

Because information on erroneous grades is crucial for the Finnish grading system, a good pairing algorithm can be adjusted to meet such requirements. With only a few games, say five, a player whose grade is too low, needs opponents closer to his real strength as early as possible that his his actual grade might be found out. For example, someone who has won three games out of three, has to get an opponent who is at least one stone stronger than the strongest player he has already beaten to find an upper limit estimate for his actual grade. When the grading distribution has large gaps, common in Finnish tournaments, care must be taken to extract grading information.

2.3.2 Other grading systems

In the United Kingdom, a mathematical model based on tournament results is used to calculate ratings for each player. These ratings are supposed to be used by the players as guidelines on which grade to use while registering for a tournament. In any case, players may use any grade they want, but the use of a grade close to their rating is strongly adviced. The mathematical model itself is based on GoR (*Go Rating*, see Section 2.2) corrected with a factor obtained from a grade/rating regression line of the European Go Federation (EGF) rating list [1].

In Germany, the players may decide their grades autonomously [10].

In the Netherlands, the kyu grades are self-assigned. The dan grades are based on tournament results, and are decided by a grading committee consisting of three strong (4 dan or higher) dan players [6].

In the Czech Republic, the player's grade is based solely on GoR. If a player's GoR reaches the nominal (20 kyu = 100 GoR, 1 kyu = 2000 GoR, 1 dan = 2100 GoR, etc.) of a grade higher than his, the player is promoted to that nominal grade. The grades are checked every three months and if a player's GoR is significantly lower (the tolerance is 100 GoR points for dan players, more for kyu players) than the nominal the player's grade is lowered according to his GoR. This leads to deflation (increase of the player's strength relative to grade) due to the GoR system's basic properties. The system, like GoR, does not adapt well enough to players improving, especially those improving fast.

Deciding the grades by themselves naturally demands lots of common sense from the players.

2.4 Go tournaments

Man, being a competitive creature by nature, wants to be in state of rivalry with other individuals of his species. Ergo, Go players organize tournaments. Of course, there is also the aspect of social networking, meeting old friends and making new ones.

Some common tournament systems are described in Chapter 3.

2.4.1 Tournament in terms of statistics

Players' strength can be modeled with discrete random variables whose parameters (mean and variance) are unknown. By using random experiments (games) some information on the means can be obtained. The exact values are impossible to solve, but with enough rounds it is possible to have a good estimates of their relative values can be found. Based on statement 1 on grades in Section 2.1, the variance decreases as the mean increases.

2.4.2 The goals of a Go tournament

A Go–or chess–tournament is usually organized to find the three top standings. Very often there is a secondary goal of collecting information about the players' strength, especially important in Go because the information will be needed in future tournaments and handicap games.

Finland is one of the few European countries without neither free grading nor a mathematical model for grading (see Section 2.1). Therefore the grading information is more important than in most other countries.

The goals of a tournament can be briefly described, using the terms presented in subsection 2.4.1, as

1. Trying to find one to five largest mean strengths.

2. Trying to put the largest mean strengths in the correct order.

3. Trying to estimate the mean strength of every player as accurately as possible.

2.4.3 Pairing problems and the criteria of a good pairing

In orded for a tournament to be informative and fun to play in, its pairing must meet certain criteria. There are a number of basic problems in pairing, part of which can be solved by choosing the tournament system and pairing method wisely.

- First of all the tournament must have a pairing for each round.
- The pairing has to give the organizers the information they want, be it the winner or the strongest player, the top five rankings, or as much grading information as possible, or – as in most cases – more than one of the above.
- A good pairing provides a *fair pairing* for the players meaning that they should get opponents as close as possible to their own strength.

Not every tournament system takes into account the non-deterministic nature of the games, i.e., the stronger player does not always win, but loses with a certain probability.

2.4.4 Social pressure versus statistics

Sometimes common sense works a bit different from the real statistics behind the tournament results. People tend to have very strong, unfounded opinions on different matters. The fact of the stronger player not necessarily winning every game induces some problems.

Let us for example take a *round robin* (see Section 3.1) tournament where two players, *A* and *B*, end up sharing the first place with equal number of wins. As stated by the rules of the round robin system, these players have played a game against each other. Let us assume that player *A* won that game. To analyze which of the players deservers to win the tournament, let us remove the game which has the most random result from the set of *A*'s and *B*'s games. Because *A* and *B* are very likely to have near equal strength, their winning probabilities are (almost) the same. Because of this, the game between *A* and *B* is the one with the most random result (its result is the most difficult to predict) and therefore should be removed. After the game is removed, *B* is left with one more win than *A* so by having more wins, he should win the tournament, which most people may find surprising.

If the results that both players have in common are removed only the game between *A* and *B* and a their games with a third player, *C*, are left. Because *A* and *B* had an equal number of wins and *B* lost to *A*, *A* lost to *C* and *C* lost to *B*. Because *A* and *B* share the top position, *C* has to have less wins than either of them and therefore be lower in the standings and probably weaker. Thus, *A* has lost to a weaker player *C* when *B* lost to an almost equal player *A*. Therefore, *A*'s result is inferior to *B*'s result.

2.4.5 Probabilities and entropy

Let us use for example GoR as the players' rating. The GoR system is based on winning probabilities. The winning probability for the weaker player, *A*, is

$P(A) = 1/(e^{D/a} + 1)$

and for the stronger, B,

$P(B) = 1 - P(A) - \epsilon,$

where ϵ is a constant describing the improvement of players, currently set at 0.016.

In short, *entropy* describes the amount of information in a (random) message based on the probabilities of message's units, most commonly bits. Entropy (and therefore the information) is maximized when the probabilities are evenly distributed.

Given a discrete random variable X with possible values $x_1, ...x_n$ and their probabilities $p(x_i)$, the entropy can be calculated with:

$$H(X) = -\sum_{i=1}^{n} p(x_i) log_b p(x_i) \tag{2.3}$$

in which the logarithm's base b defines the unit of entropy. Because we are dealing with bipolar information (wins and losses, let us ignore the draws for now), we select $b = 2$ and *bit* for our unit of entropy.

2.4.6 Probability matrix

Based on the players' ratings, let us calculate winning probabilities for each game in the Table 2.2.

Table 2.2: Players, ratings and winning probabilities

	Opponent					
Player, rating	A	B	C	D	E	F
A, 509	-	0.10126	0.56150	0.16567	0.71073	0.31841
B, 902	0.89874	-	0.91733	0.64980	0.95209	0.79604
C, 464	0.43850	0.08267	-	0.13633	0.65916	0.26894
D, 800	0.83433	0.35020	0.86367	-	0.92033	0.70893
E, 340	0.28927	0.04791	0.34084	0.07967	-	0.16415
F, 646	0.68159	0.20396	0.73106	0.29107	0.83585	-

Then calculate the entropies using equation 2.3 for each possible pairing and select the pairing resulting in maximal total entropy and thus maximal information.

Many algorithms for maximizing or minimizing sums have been developed, for example the Edmonds's "Maximum Weight Perfect Matching algorithm" [8].

Table 2.3: Entropies based on table 2.2, the chosen pairing in **bold**

Player / Opponent	A	B	C	D	E	F
A	-	0.33455	0.46753	0.42968	0.35012	**0.52571**
B	0.13843	-	0.11420	**0.40413**	0.06744	0.26197
C	0.52153	0.29732	-	0.39192	**0.39635**	0.50955
D	0.21802	**0.53011**	0.18262	-	0.11023	0.35183
E	0.51765	0.21001	**0.52927**	0.29078	-	0.42792
F	**0.37694**	0.46781	0.33039	0.51827	0.21622	-

After the round has been played, the players' ratings are updated before the next round and the next pairing.

Chapter 3

Tournament systems

There are many different tournament systems. The choice of the system must be based on the amount of players, the main goal(s) of the tournament, the amount of time available (a day, weekend, two weeks) and the nature of the sport. For example, in boxing, a cup is quite the obvious choice for medical reasons alone.

A tournament system consists of a few very basic rules. The main task of a system is to decide a *pairing* for each round. Pairing is a data set telling who are to play against each other, who are absent and who has a free round.

The most popular tournament systems in Go are presented below.

3.1 The round robin system

One could say that the maximum information without rematches can be obtained from a round robin tournament. In a round robin, every player playes exactly once against every other player. If there is an even number of players, round robin needs N-1 rounds, where N is the number of players participating. If the number of participants is odd, N rounds need to be played, because one player at a time gets a free round. The algorithm for running a round robin tournament on a long table can be stated with three rules.

1. Players take their seats for the first round

2. Player 1 stays put for the whole tournament

3. For each round, other players move to the seat on their left, skipping player 1

An example is shown in table 3.1.

Table 3.1: An example of round robin for ten players

Board	1	2	3	4	5
Player 1	**1**	2	3	4	5
Player 2	6	7	8	9	10

Round 1

Board	1	2	3	4	5
Player 1	**1**	6	2	3	4
Player 2	7	8	9	10	5

Round 2

Board	1	2	3	4	5
Player 1	**1**	7	6	2	3
Player 2	8	9	10	5	4

Round 3

Board	1	2	3	4	5
Player 1	**1**	8	7	6	2
Player 2	9	10	5	4	3

Round 4

If the number of participants is odd, a *bye* is added as a dummy player and the game against bye is considered a free round. The seating table can also be used for deciding colors. For example the upper row plays the first round with black and then alternating between white and black. Alternating is necessary because player number one always sits on the upper row.

Obviously round robin is a quite time-consuming tournament system and therefore used mostly for very small tournaments of four to six players or very fast games. When three (or more) players with equal number of wins form a cycle (*A* loses to *B*, *B* loses to *C*, and *C* loses to *A*), round robin provides no mechanism to sort the players.

3.2 Cup

Cup, also known also as “knock-out tournament” is a very simple system. The player who wins, continues in the tournament and the loser drops out. A short example tournament is shown in table 3.2. The optimal number of players is 2^N where N is the number of rounds to be played. If the number of players is not a power of two, some players, usually decided by a lottery, begin the tournament on the second round so that the number of players on the second round is a power of two.

Table 3.2: An example of a cup for eight players, winners are written in **bold**

Round 1	Round 2	Round 3	Result
Player 1	Player 2	Player 3	Player 7
Player 2			
Player 3	**Player 3**		
Player 4			
Player 5	Player 6	**Player 7**	
Player 6			
Player 7	**Player 7**		
Player 8			

Because Go games are not deterministic, i.e., the winner can not be known beforehand (like for example in a weighing competition), the strongest player does not always win a cup tournament. The tournament is (usually) organized to find the strongest player, this might be considered undesirable. Using cup to organize a Go tournament is very rare, because most of the players would play only on one or two rounds, which is not very attractive if you travel a few hundred miles to a tournament.

3.2.1 Double elimination

Double elimination works just like a cup, only that a player is dropped out after two losses. Players having won all their games play each other and the players with one loss with each other. $\log_2 N$ (rounded up) rounds are needed to find out the winner, but additional games are needed to settle the following places as there will be multiple players in the group of players having lost one of their games.

A bit better than the cup system, double elimination gives the players an extra game but the strongest player is not necessarily the winner.

3.3 The American Go Association system

The American Go Association, AGA, uses a few methods to pair their tournaments. One of them is the McMahon system (Chapter 4) with the AGA rating specific pairing routine implemented in Paul Matthews' *AccelRat*. AccelRat sorts the players by their AGA ratings and, starting from the top, pairs the closest two players who have not yet played each other.

An *"undesirability score"* consisting of rating difference, timezone difference (for Internet tournaments), and a few other criteria is calculated for each eligible player, but the rating difference has the largest effect.

After each round, the ratings are updated according to the results. If there are rounds left to be played, a new pairing is made.

Partly working like the statistics based system described in 2.4.5, the AGA system gives a lot information to the rating system. It also ensures that the players get opponents close to their own strength.

3.4 The Swiss system

Because the time available for a tournament is usually limited, round robin becomes impossible except for very fast games. Chess players often use the Swiss system, which needs less rounds to determine the winner. The number of rounds to be played is fixed before the tournament begins.

On the first round, every player starts with 0 points. Using one of the pre-determined algorithms, which are more thoroughly described in subsection 4.2.3, all the N players are paired, resulting in $N/2$ pairs in case of even N, $(N-1)/2$ in case of an odd N. After the first round, half of the players have 1 point, and the other half has 0 points. For the following rounds, the players having the same amount of points are paired with each other. The players having the same number of points are called a *group*. This way, we can determine the winner in $\log_2 N$ (rounded up) rounds.

The Swiss system can be described with two simple rules

1. Players in the same group are paired with each other. If the number of unpaired players in the group is odd, one player is paired to a player from the group having one point

less. If there is no group with less points, one player gets a free round which counts as a win.

2. Two players must not play against each other more than once. No player may get more than one free round.

The three main problems in the Swiss system are 1) how to pair so that every player gets a fair pairing, 2) if the number of players in a group is odd, who should be paired to the lower group and 3) if the number of players is big, it may take an unreasonable amount of rounds to solve the winner. The second problem is a huge one in McMahon tournaments as well. Also, as in the cup system, the winner is not necessarily the strongest player, but usually we are willing to accept this flaw to keep the number of rounds down to a reasonable number, which is the Swiss' upside.

3.5 The McMahon system (basic principles)

The McMahon system is named after Lee E. McMahon (1931-1989), a researcher at Bell Labs and an Unix developer [10]. McMahon system is also called "accelerated Swiss", because the main difference is to give players points before the first round based on their grade (see section *2.1 The grading system in Go*), to a number of rounds of Swiss having been played. The idea behind this is to reduce the amount of rounds needed and to partly solve the first problem of the Swiss system, how to pair players on the first round. The McMahon system will be covered more thoroughly in chapter 4.

Chapter 4

The McMahon system

McMahon was developed to solve the large player amount problem of the Swiss system. Dividing the players to groups by giving them points beforehand as if some rounds of Swiss had been played.

4.1 The rules for the McMahon system

1. Starting points are given to each player based on their grade or some other basis.
2. Two players must not play against each other more than once.
3. On each round, players in the same group, i.e., having the same amount of points, are paired with each other. If the number of players in a group is odd, one player is paired to a player in the next group with less points.
4. If the number of players in the round is odd, a player (usually lowest in the standings) who has not been on a free round gets a free round.
5. Players get one point for a win or a free round, half a point for a draw (*jigo* in Go) or absence, and no points for a loss.
6. A player's total points are the sum of the starting points he had and the points he gets for wins, free rounds or absence.

4.1.1 Starting points

The starting points are used to divide players into groups before the pairing on the first round. When deciding how to divide the players into groups, at least the following things must be taken into account.

- The number of players: 10, 50, 100, 700?
- The distribution of grades. Even or scattered? See Figures 4.1 and 4.2.
- The goals of the tournament. Grading information, a single winner, the best five players, or all of these?

In very large tournaments, for example the European Go Congress 2005 with 712 players in the main tournament, where the grades are more evenly distributed (Figure 4.1), the most usual way of giving starting points is to give no points for players who are 20 kyu or weaker, one point to 19 kyus and so on to 19 points to 1 kyus. 1 dan gets 20 points, 2 dan gets 21 points and so on. A few of the highest groups can be combined if there is a larger number of players capable of winning the tournament than just the players graded highest (a 4 dan can win a tournament where there are three 5 dans with a quite reasonable probability) or the number of players having the highest grade is small. For example if there is only one 5 dan player in the tournament and let us say seven 4 dans, the 5 dan gets the same amount of points as the 4 dans, because any of them could win the tournament. This is because a grade difference of only one stone does not yet affect winning probabilities very much.

Determining the players who have a real chance of winning the tournamet is a difficult task, but good results can be achieved either counting on experience or calculating the winning probabilities based on grades. The EGF tournament database gives quite a lot of information on winning probabilities between grades.

In most cases, the number of participants is quite small, under 100 players, and the grades are not distributed very evenly especially in the lower grade regions. Then the starting-point problem becomes quite hard. An example of this can be seen in the grade distribution histogram from Oulu Summer Tournament 2005 (Figure 4.2).

In these cases, the groups should be quite large at the both ends if the grade distributions allow and smaller in the middle. The highest and lowest group half their sizes every round.

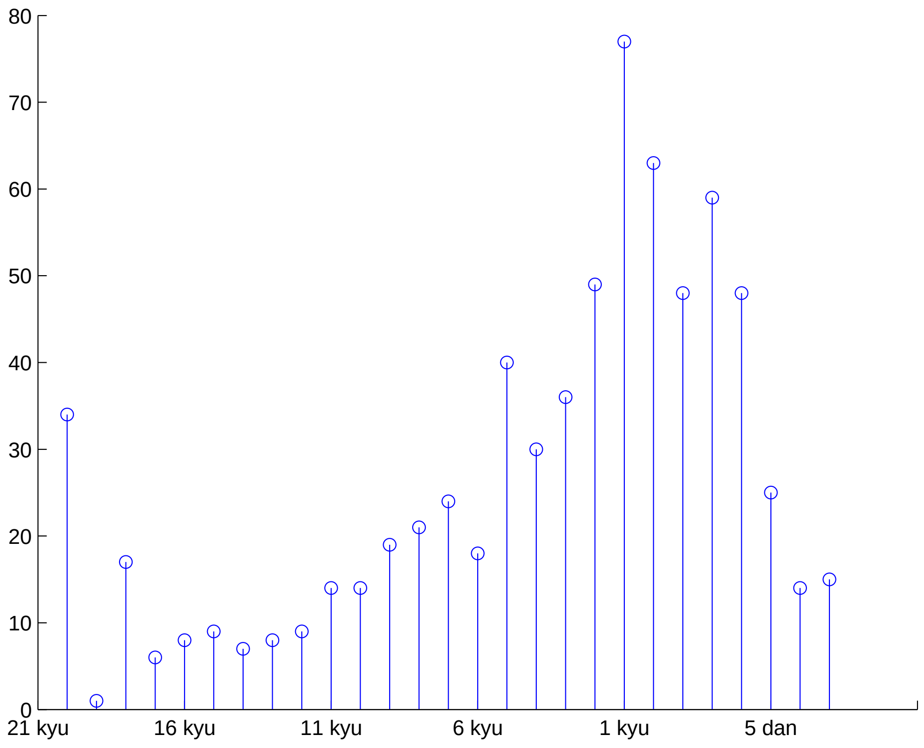


Figure 4.1: The grade distribution histogram from 21 kyu to 7 dan at European Go Congress 2005

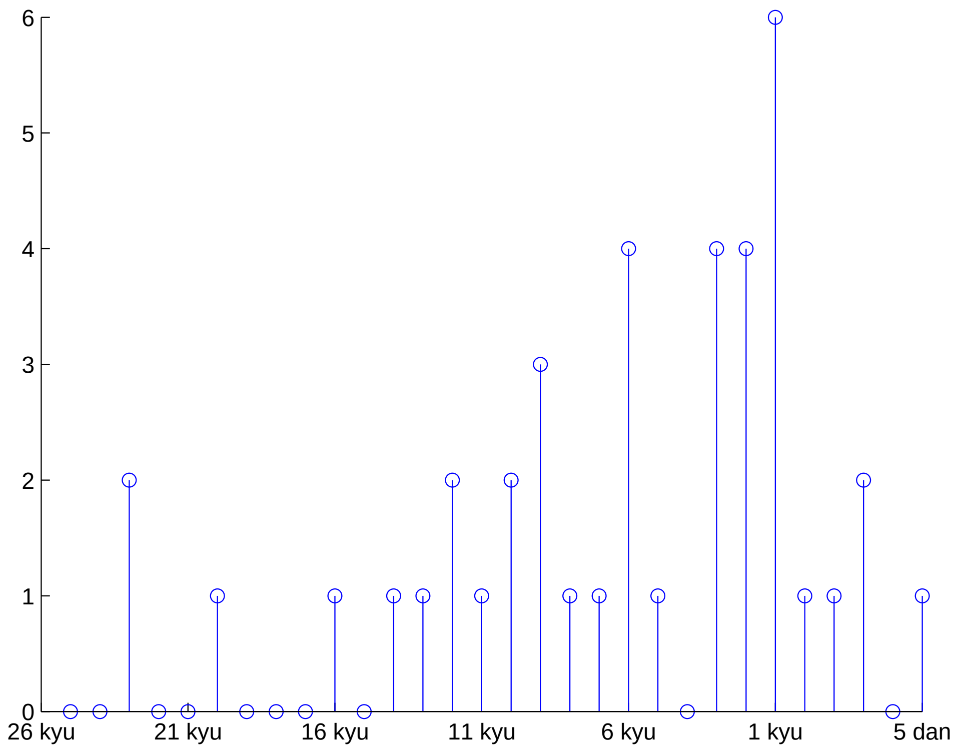


Figure 4.2: The grade distribution histogram from 26 kyu to 5 dan at Oulu Summer Tournament 2005

Table 4.1: Usual group sizes in a five-round tournament

Players	10-20	15-30	25-50	45-70	65-90	85-110	100-150	140-200	170-
Groups	1-4	3-6	5-8	6-10	9-12	10-15	11-20	12-25	20-

The total number of groups depends mainly on the number of players. If the range of the grades is very narrow, fewer groups are needed. An example of usual group sizes can be found in Table 4.1.

4.2 The mechanics

4.2.1 Scoring

In McMahon, a player gets one point for each win or free round, no points for a loss and half a point for a draw. Giving half a point for absence and rounding the points gained this way down to avoid groups with only one player is also used.

The McMahon score (*MMS*) is the total sum of a player's starting points, wins and the points acquired from free rounds and absence.

4.2.2 Secondary placement criteria

Because most of the players end up in a group with other players with the same MMS, to decide standings and to give more information to the pairing algorithm, secondary placement criteria, also known as tie-breakers, have to be used. There are lots of options. The final decision of which to use is totally up to the tournament organizers.

Wins

The number of games the player has won. The players in the same group are sorted according to the groups they started from, lowest starting group first.

SOS

SOS stands for *Sum of Opponents' Scores*. A player's SOS is calculated by summing together all the player's opponents' MMS. On a free round, when the players has no opponent, SOS is increased by the player's own MMS. The score describes how strong oppponents the player has had.

SOSOS

Standing for *Sum of Opponents' SOS*, SOSOS is often used as the secondary tie breaker after SOS. It describes how strong opponents the player's opponents have had.

SODOS

Sum Of Defeated Opponents' Scores is calculated by summing the points of the player's opponents he has won. It is usually used after SOS. SODOS tells how strong opponents the player has defeated.

CUSS

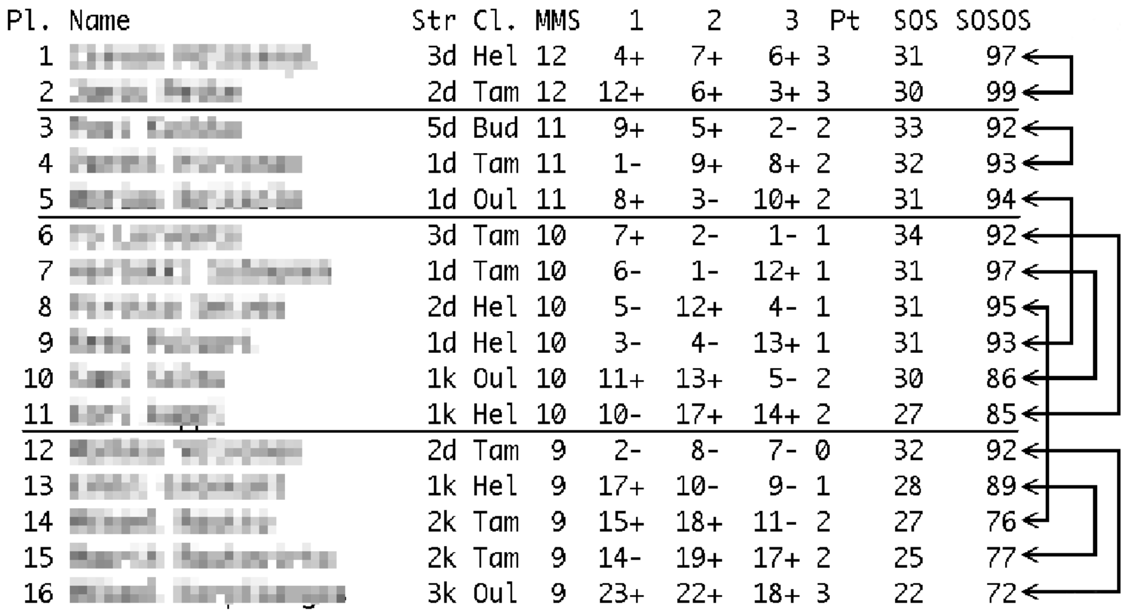
CUSS stands for *CUmulative Sum of Scores*.

$\sum_{k=1}^{n} S(k)$,

where n is the number of rounds and $S(k)$ is the amount of McMahon points after round k. Gives more points for winning in the beginning of the tournament than in the end. For example if a player wins his first three games out of five (and has no starting points), CUSS is $1 + 2 + 3 + 3 + 3 = 12$ and if the three wins accour on the last three rounds (i.e., after losing two games and ending up to play with weaker opponents), $0 + 0 + 1 + 2 + 3 = 6$.

4.2.3 Pairing inside a McMahon group

One of the biggest unsolved problems in the Swiss system (and therefore also in the McMahon system) is "how to pair inside a group". Most of the methods are based on players' SOS or other numeric information calculated from the results of previous rounds. This is called



Pl.	Name	Str	Cl.	MMS	1	2	3	Pt	SOS	SOSOS
1	[redacted]	3d	Hel	12	4+	7+	6+	3	31	97
2	[redacted]	2d	Tam	12	12+	6+	3+	3	30	99
3	[redacted]	5d	Bud	11	9+	5+	2-	2	33	92
4	[redacted]	1d	Tam	11	1-	9+	8+	2	32	93
5	[redacted]	1d	Oul	11	8+	3-	10+	2	31	94
6	[redacted]	3d	Tam	10	7+	2-	1-	1	34	92
7	[redacted]	1d	Tam	10	6-	1-	12+	1	31	97
8	[redacted]	2d	Hel	10	5-	12+	4-	1	31	95
9	[redacted]	1d	Hel	10	3-	4-	13+	1	31	93
10	[redacted]	1k	Oul	10	11+	13+	5-	2	30	86
11	[redacted]	1k	Hel	10	10-	17+	14+	2	27	85
12	[redacted]	2d	Tam	9	2-	8-	7-	0	32	92
13	[redacted]	1k	Hel	9	17+	10-	9-	1	28	89
14	[redacted]	2k	Tam	9	15+	18+	11-	2	27	76
15	[redacted]	2k	Tam	9	14-	19+	17+	2	25	77
16	[redacted]	3k	Oul	9	23+	22+	18+	3	22	72

Figure 4.3: Folding

"seeding". The methods used most often are described below. The lowest group in Figures 4.3 to 4.5 demonstrates the method in question the best because the upper groups are a bit more difficult to pair because of lots of pairings from one group to another.

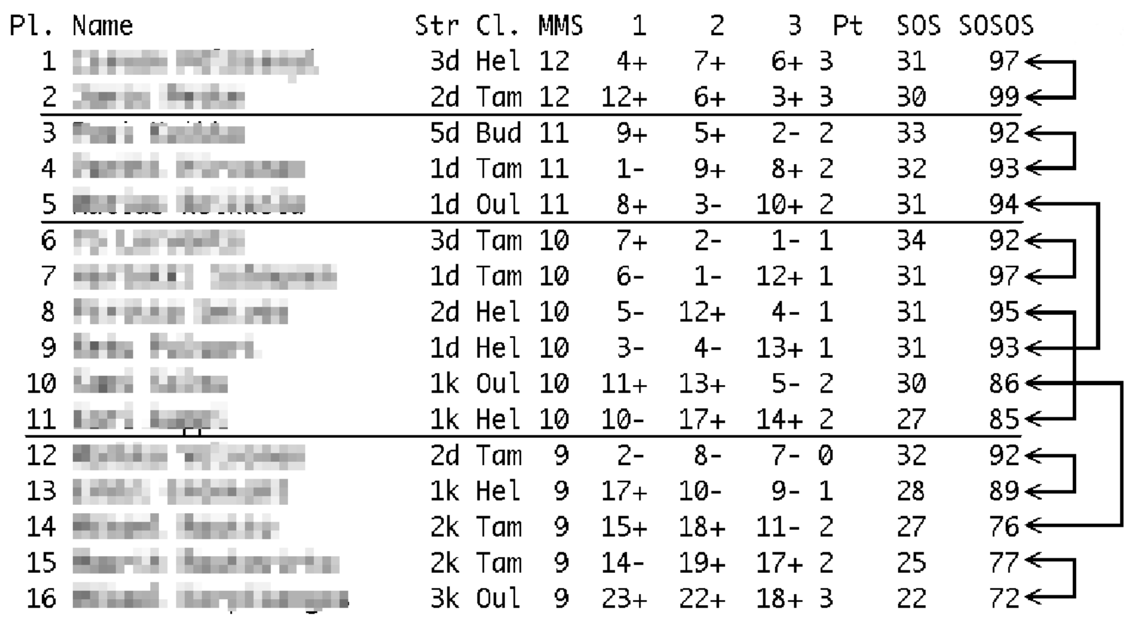
The example figures may appear a bit messy because they are not "clean" textbook examples but taken from a real tournament. Because of this, there are pairings from one group to another that make the algorithms a bit unclear.

Folding

In this method the group is folded, the upper half is turned upside down and laid over the lower half. The player(s) to be paired up or down is/are in the middle of the group. For example in a group of eight players, number 1 is paired to number 8, 2 to 7 et cetera. If the players in the group are numbered from 1 to N from top to bottom, $pair(x) = N - x + 1$. See Figure 4.3.

Double folding

Double folding is based on dividing the group in three parts: The upper quarter, middle half and the lower quarter. The upper and lower quarters are flipped upside down and laid over the middle half. For example in a group of 16 players player number 1 is paired to player



Pl.	Name	Str	Cl.	MMS	1	2	3	Pt	SOS	SOSOS
1	[redacted]	3d	Hel	12	4+	7+	6+	3	31	97
2	[redacted]	2d	Tam	12	12+	6+	3+	3	30	99
3	[redacted]	5d	Bud	11	9+	5+	2-	2	33	92
4	[redacted]	1d	Tam	11	1-	9+	8+	2	32	93
5	[redacted]	1d	Oul	11	8+	3-	10+	2	31	94
6	[redacted]	3d	Tam	10	7+	2-	1-	1	34	92
7	[redacted]	1d	Tam	10	6-	1-	12+	1	31	97
8	[redacted]	2d	Hel	10	5-	12+	4-	1	31	95
9	[redacted]	1d	Hel	10	3-	4-	13+	1	31	93
10	[redacted]	1k	Oul	10	11+	13+	5-	2	30	86
11	[redacted]	1k	Hel	10	10-	17+	14+	2	27	85
12	[redacted]	2d	Tam	9	2-	8-	7-	0	32	92
13	[redacted]	1k	Hel	9	17+	10-	9-	1	28	89
14	[redacted]	2k	Tam	9	15+	18+	11-	2	27	76
15	[redacted]	2k	Tam	9	14-	19+	17+	2	25	77
16	[redacted]	3k	Oul	9	23+	22+	18+	3	22	72

Figure 4.4: Double folding

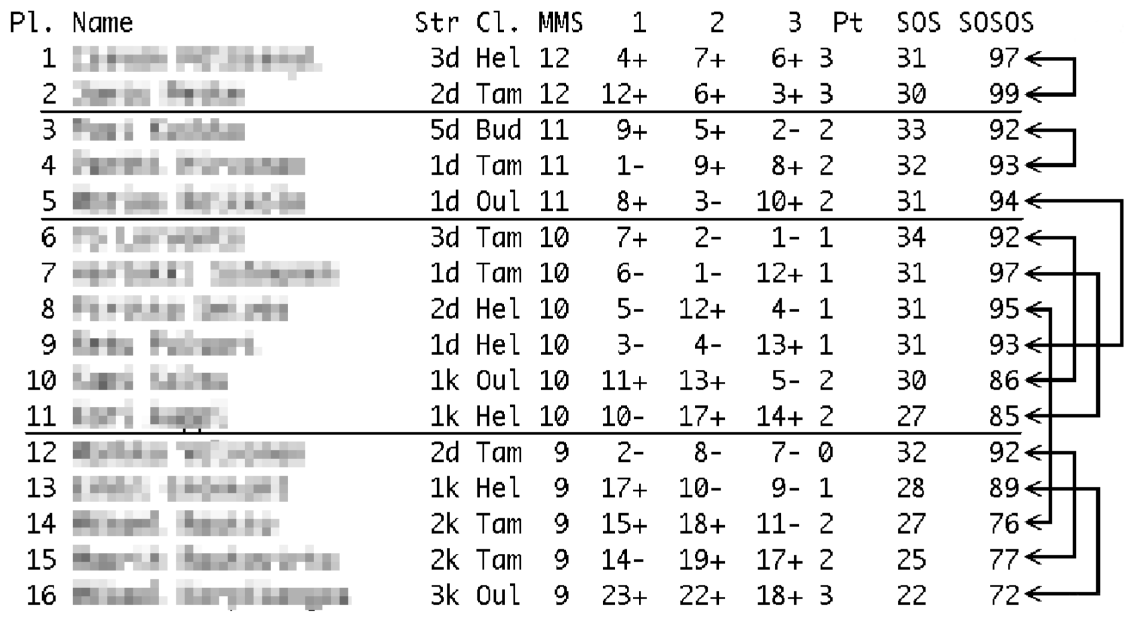
number 8, 2 to 7 et cetera and player number 16 is paired to player number 9, 15 to 10 etc. The player to be paired to a lower group is selected from the middle of the group. In the example data one can see that this is not always possible if for example players have played against each other already. With players numbered $1..N$, $pair(x) = N - x + 1$. See Figure 4.4.

Translation

In this approach the upper half of the group is shifted on the lower half of the group. For example with a eight player group, player number 1 is paired with player number 5, 2 to 6 et cetera. If the players in the group are numbered from 1 to N, $pair(x) = x + \frac{N}{2}$. See Figure 4.5.

Random

Using random pairing is an intuitive way to approach this problem. One could say that on the average everyone gets a fair pairing. It is probable that some players will not get a fair pairing, because the number of rounds in a tournament is insufficient to achieve a number of samples large enough to get very close to the average.



Pl.	Name	Str	Cl.	MMS	1	2	3	Pt	SOS	SOSOS
1	[redacted]	3d	Hel	12	4+	7+	6+	3	31	97
2	[redacted]	2d	Tam	12	12+	6+	3+	3	30	99
3	[redacted]	5d	Bud	11	9+	5+	2-	2	33	92
4	[redacted]	1d	Tam	11	1-	9+	8+	2	32	93
5	[redacted]	1d	Oul	11	8+	3-	10+	2	31	94
6	[redacted]	3d	Tam	10	7+	2-	1-	1	34	92
7	[redacted]	1d	Tam	10	6-	1-	12+	1	31	97
8	[redacted]	2d	Hel	10	5-	12+	4-	1	31	95
9	[redacted]	1d	Hel	10	3-	4-	13+	1	31	93
10	[redacted]	1k	Oul	10	11+	13+	5-	2	30	86
11	[redacted]	1k	Hel	10	10-	17+	14+	2	27	85
12	[redacted]	2d	Tam	9	2-	8-	7-	0	32	92
13	[redacted]	1k	Hel	9	17+	10-	9-	1	28	89
14	[redacted]	2k	Tam	9	15+	18+	11-	2	27	76
15	[redacted]	2k	Tam	9	14-	19+	17+	2	25	77
16	[redacted]	3k	Oul	9	23+	22+	18+	3	22	72

Figure 4.5: Translation

4.2.4 Free rounds

If the number of players on a round in the tournament is odd, one player has to get a free round. Usually this is the player lowest in the standings who has not been on a free round yet. Free round increases the player's points by one and SOS (see 4.2.2) by players own points.

4.3 Problems of the McMahon system

There are a few informational gaps left by the definition of the McMahon system. First of all, how to determine the starting points. The system leaves this anything-but-easy problem to the tournament organizers to decide. This problem must be approached with the goals of the tournament in mind. How many prizes are there to give, how much grading information is needed and can it be obtained with the players "available", and so on.

Also unspecified by the system, as in the Swiss system, is the way the players inside one group should be paired. Also the players paired to another group present a problem, maybe the most difficult one. All of these decisions affect the players' secondary placement criteria and can cause quite a big problem especially among the winner candidates because determining the winner depends on the pairing mechanics.

Most McMahon programs aim to even the SOS among the players in the same group by pairing down the player with the largest SOS. This method is quite good and intuitively acceptable, but the functionality of SOS as a tie-breaker is lost. If the pairing is based on SOS this way, it should not be used as a tie-breaker at the same time because these are two opposite goals. SOS as a tie-breaker is based on differences in SOS. If the pairing algorithm tries to minimize these differences, it renders the information we get from SOS useless.

If SOS is used as a tie-breaker, the player with the largest SOS is at the top of a group of say X points. If he gets paired down and all the other players of the group inside the group, half of the other players (those who won their game) get X points SOS, the other half (those who lost) $X + 1$ and the player paired down gets $X - 1$. This way, the SOS differences between the players get smaller.

Of course, other tie-breakers suffer from the same problems as SOS because of the very small amount of available information. For example, five games per player gives a very low probability of getting a sufficient amount of information. Pairing downward has a negative impact on all strength estimators, especially SOS.

When trying to obtain as much grading information as possible, players who have won all their games should be paired to someone significantly, but not too much stronger. Too large grade differences lead to games with too predictable results while maximum information is achieved from games with winning probabilities differing only slightly from 0.5, (see 2.4.5) which does not apply to obtaining the maximum change in GoR like calculated in 2.2.2.

This is why pairing down from the top of the group and up from the bottom of the group, as when trying to even SOS points, is bad for grading information especially in tournaments with scattered grading distribution. If a player has won most of his games, and therefore has been positioned at the bottom of his group, he should be paired against opponents that are one to three stones stronger than the ones they have already won.

For example a 6 kyu player with a result line *6k+ 5k+ 5k+* should get maybe a 3 kyu who has won at least one of his three games, not a shodan like in a lot of cases with poor grading distribution, to achieve 1) more grading information and 2) some confidence to the 3 kyu's grade. If the 6 kyu loses, his next game should be for example against a 4 kyu with a 50% result line.

Few tournament management programs tend to pair the highest player from the group (who has lost most of his games) downwards if the number of players in the group is odd. The player to be paired up is the lowest in the next group who has won (if not all but at least) most of his games and seems to have a too low grade. Unfortunately, at least in Finland this leads to a game with a far too big grading difference and no usable grading information is obtained.

Chapter 5

The most common tournament management programs

5.1 MacMahon

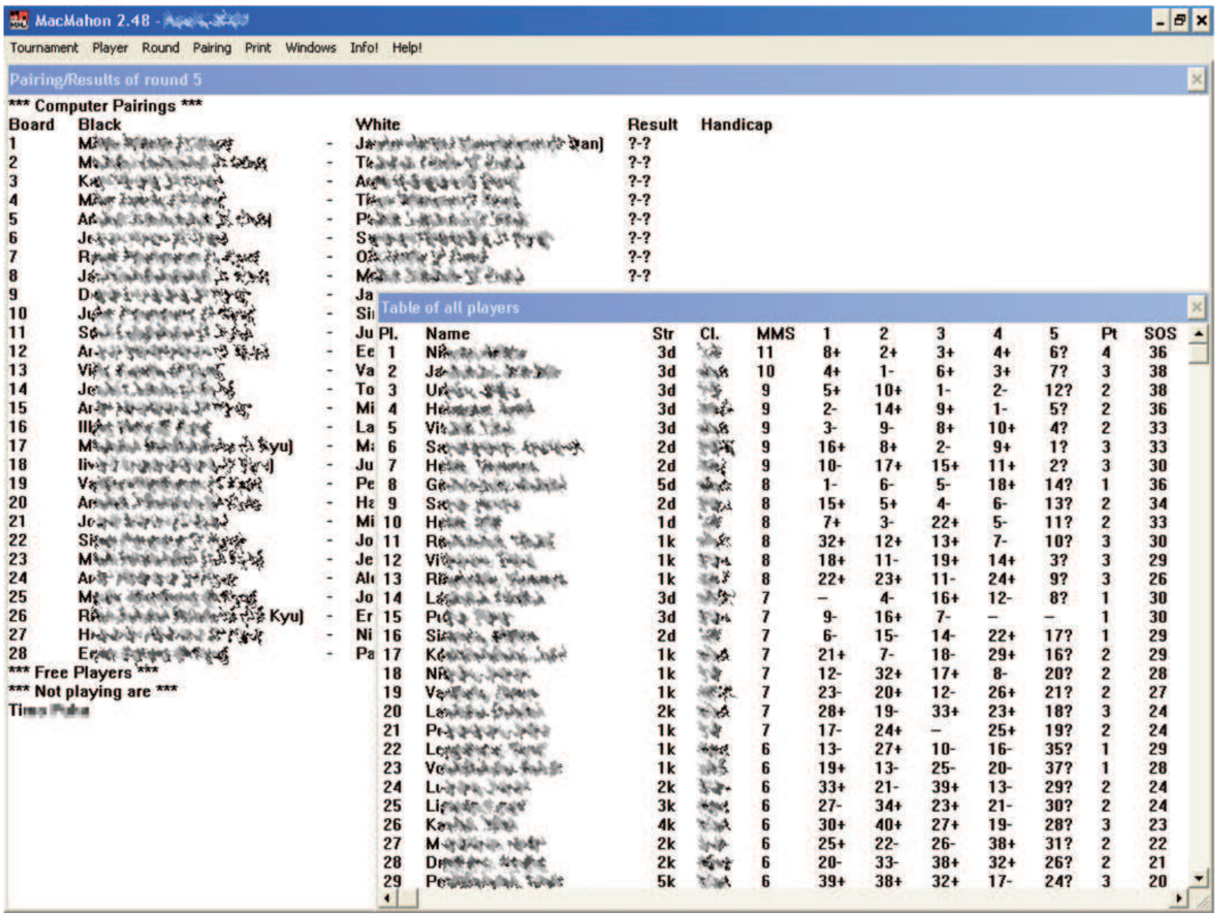
MacMahon (sic), seen in Figure 5.1, is a tournament management software written by Christoph Gerlach. The program is described in more detail in his M.Sc. thesis [5].

5.1.1 Main principles

MacMahon uses a weighting function to evaluate each potential pair. The players are then inserted into a weighted graph as nodes and the lines between the nodes are weighted with the weighting function. This graph is then reduced into $N/2$ (where N is the number of players to be paired) graphs of only two nodes (players) by Edmonds' "Maximum Weight Perfect Matching algorithm" [8]. This way, the decisions affecting the pairing are within the weighting function.

5.1.2 Evaluation

For large tournaments with a continuous grade distribution, MacMahon performs very well. In smaller tournaments pairing between groups may introduce some problems, but that unfortunately is the case with every group based pairing method.


MacMahon 2.48 - [redacted]
Tournament Player Round Pairing Print Windows Info! Help!
Pairing/Results of round 5
Table of all players

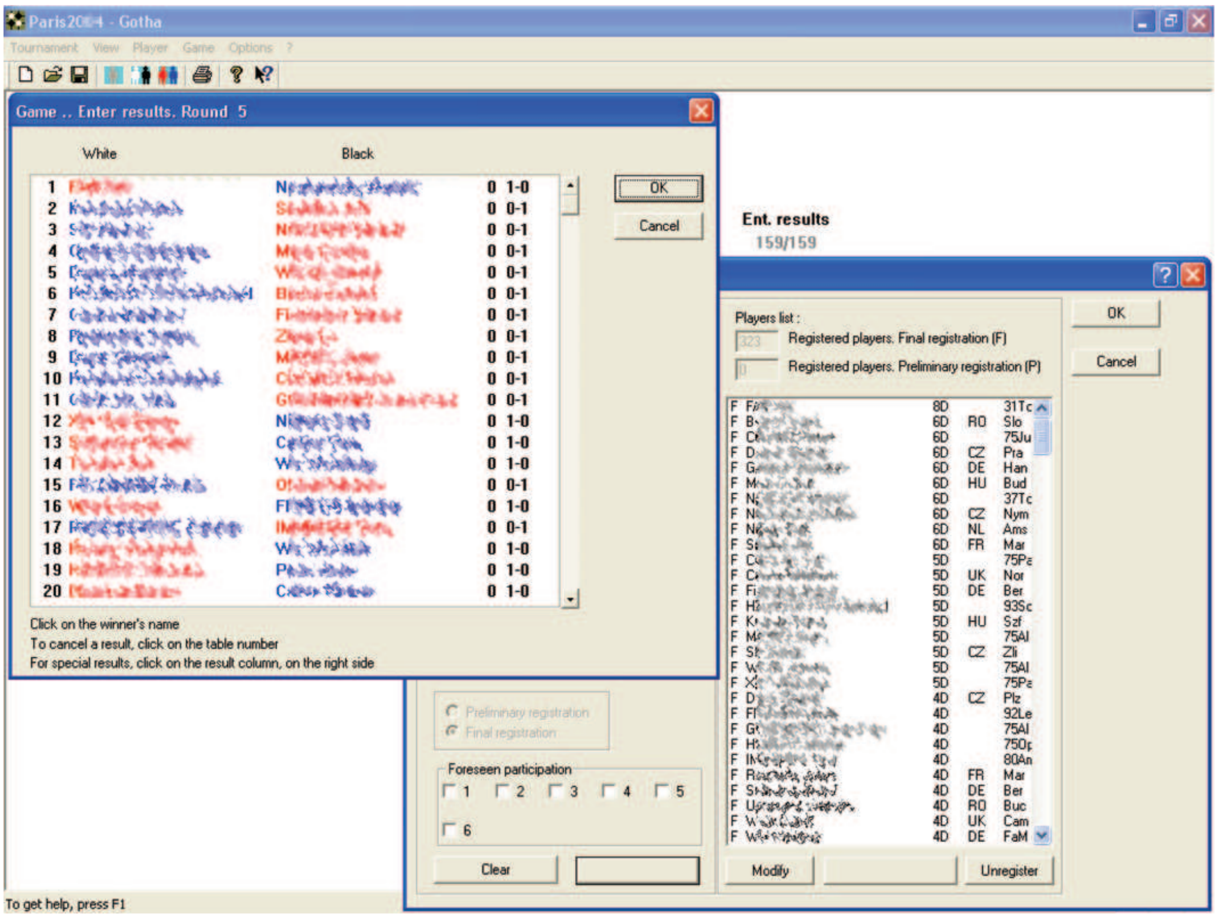
*** Computer Pairings ***

Board	Black		White	Result	Handicap
1	[redacted]	-	[redacted] Dan]	?-?	
2	[redacted]	-	[redacted]	?-?	
3	[redacted]	-	[redacted]	?-?	
4	[redacted]	-	[redacted]	?-?	
5	[redacted]	-	[redacted]	?-?	
6	[redacted]	-	[redacted]	?-?	
7	[redacted]	-	[redacted]	?-?	
8	[redacted]	-	[redacted]	?-?	
9	[redacted]	-	Ja		
10	[redacted]	-	Si		
11	[redacted]	-	Ju		
12	[redacted]	-	Ee		
13	[redacted]	-	Va		
14	[redacted]	-	To		
15	[redacted]	-	Mi		
16	[redacted]	-	La		
17	[redacted] Kyu]	-	Ma		
18	[redacted]	-	Ju		
19	[redacted]	-	Pe		
20	[redacted]	-	Ha		
21	[redacted]	-	Mi		
22	[redacted]	-	Jo		
23	[redacted]	-	Je		
24	[redacted]	-	Al		
25	[redacted]	-	Jo		
26	[redacted] Kyu]	-	Er		
27	[redacted]	-	Ni		
28	[redacted]	-	Pa		

*** Free Players ***
*** Not playing are ***
Tim [redacted]

Pl.	Name	Str	Cl.	MMS	1	2	3	4	5	Pt	SOS
1	[redacted]	3d	[redacted]	11	8+	2+	3+	4+	6?	4	36
2	[redacted]	3d	[redacted]	10	4+	1-	6+	3+	7?	3	38
3	[redacted]	3d	[redacted]	9	5+	10+	1-	2-	12?	2	38
4	[redacted]	3d	[redacted]	9	2-	14+	9+	1-	5?	2	36
5	[redacted]	3d	[redacted]	9	3-	9-	8+	10+	4?	2	33
6	[redacted]	2d	[redacted]	9	16+	8+	2-	9+	1?	3	33
7	[redacted]	2d	[redacted]	9	10-	17+	15+	11+	2?	3	30
8	[redacted]	5d	[redacted]	8	1-	6-	5-	18+	14?	1	36
9	[redacted]	2d	[redacted]	8	15+	5+	4-	6-	13?	2	34
10	[redacted]	1d	[redacted]	8	7+	3-	22+	5-	11?	2	33
11	[redacted]	1k	[redacted]	8	32+	12+	13+	7-	10?	3	30
12	[redacted]	1k	[redacted]	8	18+	11-	19+	14+	3?	3	29
13	[redacted]	1k	[redacted]	8	22+	23+	11-	24+	9?	3	26
14	[redacted]	3d	[redacted]	7	–	4-	16+	12-	8?	1	30
15	[redacted]	3d	[redacted]	7	9-	16+	7-	–	–	1	30
16	[redacted]	2d	[redacted]	7	6-	15-	14-	22+	17?	1	29
17	[redacted]	1k	[redacted]	7	21+	7-	18-	29+	16?	2	29
18	[redacted]	1k	[redacted]	7	12-	32+	17+	8-	20?	2	28
19	[redacted]	1k	[redacted]	7	23-	20+	12-	26+	21?	2	27
20	[redacted]	2k	[redacted]	7	28+	19-	33+	23+	18?	3	24
21	[redacted]	1k	[redacted]	7	17-	24+	–	25+	19?	2	24
22	[redacted]	1k	[redacted]	6	13-	27+	10-	16-	35?	1	29
23	[redacted]	1k	[redacted]	6	19+	13-	25-	20-	37?	1	28
24	[redacted]	2k	[redacted]	6	33+	21-	39+	13-	29?	2	24
25	[redacted]	3k	[redacted]	6	27-	34+	23+	21-	30?	2	24
26	[redacted]	4k	[redacted]	6	30+	40+	27+	19-	28?	3	23
27	[redacted]	2k	[redacted]	6	25+	22-	26-	38+	31?	2	22
28	[redacted]	2k	[redacted]	6	20-	33-	38+	32+	26?	2	21
29	[redacted]	5k	[redacted]	6	39+	38+	32+	17-	24?	3	20

Figure 5.1: MacMahon main view with pairing and results windows visible



	White	Black		
1	[illegible]	[illegible]	0	1-0
2	[illegible]	[illegible]	0	0-1
3	[illegible]	[illegible]	0	0-1
4	[illegible]	[illegible]	0	0-1
5	[illegible]	[illegible]	0	0-1
6	[illegible]	[illegible]	0	0-1
7	[illegible]	[illegible]	0	0-1
8	[illegible]	[illegible]	0	0-1
9	[illegible]	[illegible]	0	0-1
10	[illegible]	[illegible]	0	0-1
11	[illegible]	[illegible]	0	0-1
12	[illegible]	[illegible]	0	1-0
13	[illegible]	[illegible]	0	1-0
14	[illegible]	[illegible]	0	1-0
15	[illegible]	[illegible]	0	0-1
16	[illegible]	[illegible]	0	1-0
17	[illegible]	[illegible]	0	0-1
18	[illegible]	[illegible]	0	1-0
19	[illegible]	[illegible]	0	1-0
20	[illegible]	[illegible]	0	1-0

Click on the winner's name
To cancel a result, click on the table number
For special results, click on the result column, on the right side

Players list :
323 Registered players. Final registration (F)
0 Registered players. Preliminary registration (P)

F	[illegible]	8D		31Tc
F	[illegible]	6D	RO	Slo
F	[illegible]	6D		75Ju
F	[illegible]	6D	CZ	Pra
F	[illegible]	6D	DE	Han
F	[illegible]	6D	HU	Bud
F	[illegible]	6D		37Tc
F	[illegible]	6D	CZ	Nym
F	[illegible]	6D	NL	Ams
F	[illegible]	6D	FR	Mar
F	[illegible]	5D		75Pa
F	[illegible]	5D	UK	Nor
F	[illegible]	5D	DE	Ber
F	[illegible]	5D		93Sc
F	[illegible]	5D	HU	Szf
F	[illegible]	5D		75Al
F	[illegible]	5D	CZ	Zli
F	[illegible]	5D		75Al
F	[illegible]	5D		75Pa
F	[illegible]	4D	CZ	Plz
F	[illegible]	4D		92Le
F	[illegible]	4D		75Al
F	[illegible]	4D		75Op
F	[illegible]	4D		80An
F	[illegible]	4D	FR	Mar
F	[illegible]	4D	DE	Ber
F	[illegible]	4D	RO	Buc
F	[illegible]	4D	UK	Cam
F	[illegible]	4D	DE	FaM

Figure 5.2: Gotha main view with pairing/result entry and registration windows visible

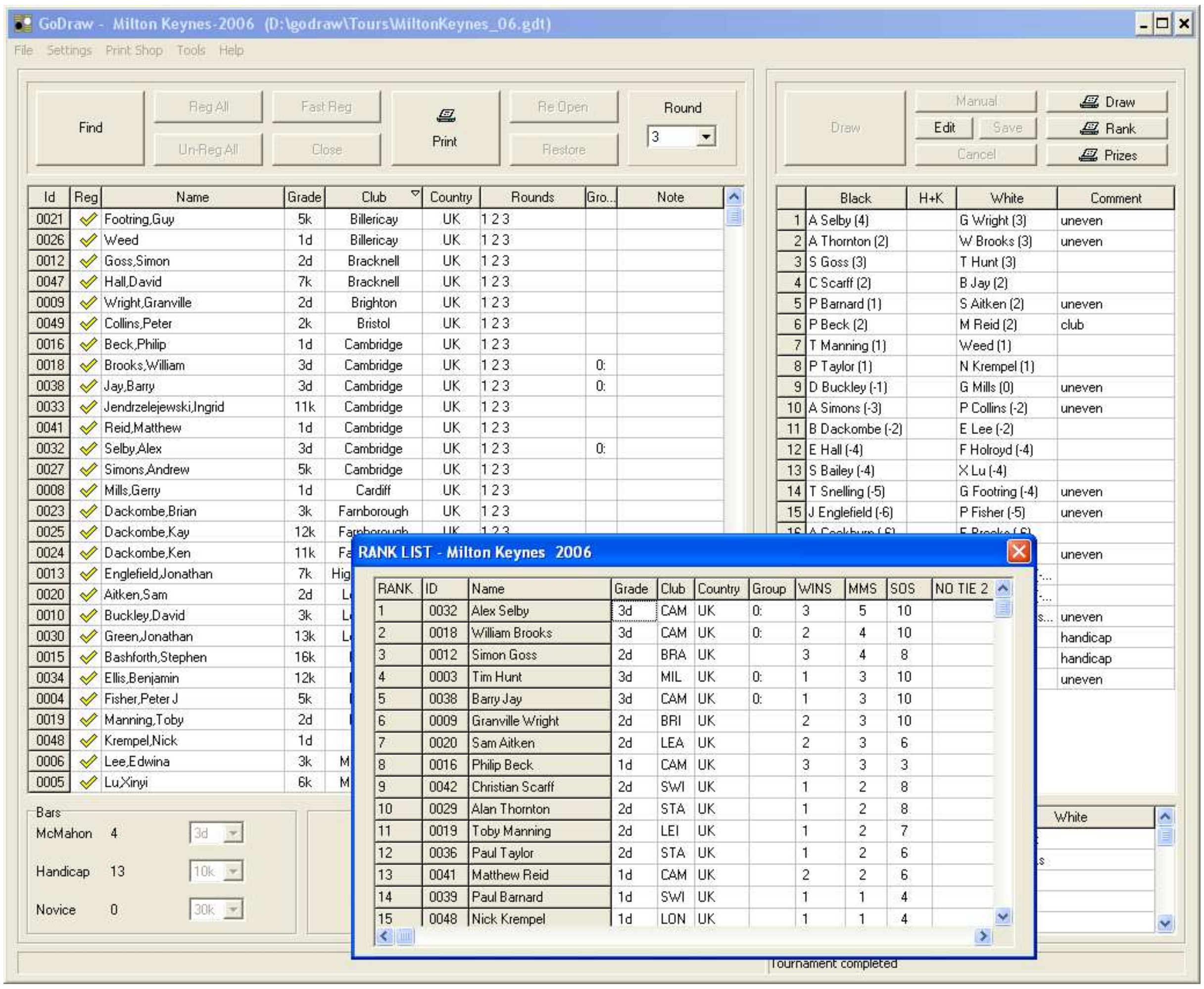
Unfortunately, only some of the parameters of the function are user-definable in the software. The user cannot affect for example the point in the group where the player to be paired up or down is selected from. This is not a flaw in the system itself, but in the user interface. Seems like this is a conscious choice to limit the user's options.

5.2 Gotha

Gotha, seen in Figure 5.2, is an open source tournament management program, used mainly in France.

5.2.1 Main principles

Like MacMahon, Gotha uses the Edmonds' algorithm. The calculation of the weights is a bit different from Gerlach's version, but very much based on the same attributes. The



Round: 3

Id	Reg	Name	Grade	Club	Country	Rounds	Gro...	Note
0021	✓	Footring,Guy	5k	Billericay	UK	1 2 3		
0026	✓	Weed	1d	Billericay	UK	1 2 3		
0012	✓	Goss,Simon	2d	Bracknell	UK	1 2 3		
0047	✓	Hall,David	7k	Bracknell	UK	1 2 3		
0009	✓	Wright,Granville	2d	Brighton	UK	1 2 3		
0049	✓	Collins,Peter	2k	Bristol	UK	1 2 3		
0016	✓	Beck,Philip	1d	Cambridge	UK	1 2 3		
0018	✓	Brooks,William	3d	Cambridge	UK	1 2 3	0:	
0038	✓	Jay,Barry	3d	Cambridge	UK	1 2 3	0:	
0033	✓	Jendrzelejewski,Ingrid	11k	Cambridge	UK	1 2 3		
0041	✓	Reid,Matthew	1d	Cambridge	UK	1 2 3		
0032	✓	Selby,Alex	3d	Cambridge	UK	1 2 3	0:	
0027	✓	Simons,Andrew	5k	Cambridge	UK	1 2 3		
0008	✓	Mills,Gerry	1d	Cardiff	UK	1 2 3		
0023	✓	Dackombe,Brian	3k	Farnborough	UK	1 2 3		
0025	✓	Dackombe,Kay	12k	Farnborough	UK	1 2 3		
0024	✓	Dackombe,Ken	11k	Fa				
0013	✓	Englefield,Jonathan	7k	Hig				
0020	✓	Aitken,Sam	2d	L				
0010	✓	Buckley,David	3k	L				
0030	✓	Green,Jonathan	13k	L				
0015	✓	Bashforth,Stephen	16k					
0034	✓	Ellis,Benjamin	12k					
0004	✓	Fisher,Peter J	5k					
0019	✓	Manning,Toby	2d					
0048	✓	Krempel,Nick	1d					
0006	✓	Lee,Edwina	3k	M				
0005	✓	Lu,Xinyi	6k	M				

Bars

McMahon	4	3d
Handicap	13	10k
Novice	0	30k

	Black	H+K	White	Comment
1	A Selby (4)		G Wright (3)	uneven
2	A Thornton (2)		W Brooks (3)	uneven
3	S Goss (3)		T Hunt (3)	
4	C Scarff (2)		B Jay (2)	
5	P Barnard (1)		S Aitken (2)	uneven
6	P Beck (2)		M Reid (2)	club
7	T Manning (1)		Weed (1)	
8	P Taylor (1)		N Krempel (1)	
9	D Buckley (-1)		G Mills (0)	uneven
10	A Simons (-3)		P Collins (-2)	uneven
11	B Dackombe (-2)		E Lee (-2)	
12	E Hall (-4)		F Holroyd (-4)	
13	S Bailey (-4)		X Lu (-4)	
14	T Snelling (-5)		G Footring (-4)	uneven
15	J Englefield (-6)		P Fisher (-5)	uneven
16	A Cockburn (-6)		F Brooke (-6)	
				uneven
			-...	
			-...	
			s...	uneven
				handicap
				handicap
				uneven

RANK LIST - Milton Keynes 2006

RANK	ID	Name	Grade	Club	Country	Group	WINS	MMS	SOS	NO TIE 2
1	0032	Alex Selby	3d	CAM	UK	0:	3	5	10	
2	0018	William Brooks	3d	CAM	UK	0:	2	4	10	
3	0012	Simon Goss	2d	BRA	UK		3	4	8	
4	0003	Tim Hunt	3d	MIL	UK	0:	1	3	10	
5	0038	Barry Jay	3d	CAM	UK	0:	1	3	10	
6	0009	Granville Wright	2d	BRI	UK		2	3	10	
7	0020	Sam Aitken	2d	LEA	UK		2	3	6	
8	0016	Philip Beck	1d	CAM	UK		3	3	3	
9	0042	Christian Scarff	2d	SWI	UK		1	2	8	
10	0029	Alan Thornton	2d	STA	UK		1	2	8	
11	0019	Toby Manning	2d	LEI	UK		1	2	7	
12	0036	Paul Taylor	2d	STA	UK		1	2	6	
13	0041	Matthew Reid	1d	CAM	UK		2	2	6	
14	0039	Paul Barnard	1d	SWI	UK		1	1	4	
15	0048	Nick Krempel	1d	LON	UK		1	1	4	

Figure 5.3: GoDraw with player list, draw (pairing) for round 3 and results visible

weights can be adjusted quite freely which also delivers a new kind of problem. The people responsible for the pairing can make the algorithm behave very strangely if they do not know what they are doing. For example changing the weighting of grade differences could lead to very uneven games. A more complex example could be the pairing between groups, pairing from the top of the upper group to the bottom of the lower group may not be a good idea if SOS is used as a tie-breaker, because the pairing evens the SOS differences.

5.3 GoDraw

GoDraw is developed by Geoff Kaniuk and used mainly in Great Britain. This chapter is based on my discussion with Mr Kaniuk in December 2005. The user interface of the software is shown in Figure 5.3.

5.3.1 Main principles

GoDraw uses a method which could be called “guided random”. Mr Kaniuk does not believe that seeding (pairing based on, e.g., SOS) helps to improve the pairings. The “random” part being quite obvious, the “guiding” is done before picking players to be paired randomly by dividing players into subgroups to obtain a better result than by using totally random pairs.

Handling groups with odd number of players

At the beginning of the pairing process, GoDraw first deals with groups with odd number of players, starting from the topmost group. The players to pair down are usually the ones who have lost their previous game. The players to pair up are the ones who have been paired down and won.

```
1:  repeat
2:      Select the next group (at first, the topmost group)
3:      if the number of unpaired players is odd then
4:          Create a subgroup of players to pair down
5:          Select a player from the subgroup by random
6:          From the next group, create a subgroup of players to pair up. For example a player
            who was paired down and won on the previous round, belongs to this subgroup.
7:          Select a player from this subgroup by random
8:          Pair the players picked on lines 5 and 7.
9:      end if
10: until no groups left
```

Pairing inside a group

To prevent players from the same country being paired, GoDraw divides the group’s unpaired players in to subgroups by country. Subgroups are then sorted by the number of players in descending order. Then a pairing is made between players from the largest two groups, picking a player at random from the largest one first. As long as there are unpaired players, the subgroups are sorted again by size, a pair picked and so on.

To prevent players from the same club being paired, a similar mechanism is used. The players from the same country are divided in to groups by clubs, the groups are sorted by the number of players in descending order etc.

The colors for the players are decided while finding the opponent for the player picked first. Based on the colors the player had on previous rounds, we know which color he should play with. If the player has played the same number of games on both colors, his favored color is decided at random. The opponent is then picked from a subgroup (of a subgroup of a subgroup...) of players favoring the opposite color.

Other things worth mentioning

To make pairing the bottom groups easier and to ensure that the methods described above work better, handicap games are allowed with even quite large (though reduced) handicaps. Because of this, grade difference is not taken into account in the pairing. According to G. Kaniuk, the players are far more satisfied with handicap games than playing their neighbour in a tournament they have travelled hundreds of miles for. On this, I agree.

5.3.2 Possible problems

The approach used by GoDraw actually works quite well. Unfortunately, the method does not work that well in Finnish tournaments. First of all, almost all Finnish tournaments nowadays are played with even games only. In 2005, 12 out of 17 Finnish tournaments were played without handicaps [11]. City championships, also played without handicaps, were left out of these statistics because they are not open to all players and the Finnish Championship consisting of four sub-tournaments was counted as one. In 2006 the corresponding figure was 14 out of 17 tournaments. As seen in Figure 5.4, the ratio has radically changed in 2002-2003.

Second, because the grading system in Finland is not based on any numerical system, but on stronger people deciding the weaker players' grades, the GoDraw gives far too little of grading information. A player winning his games has to get stronger opponents and preferably also early so that his grade can be approximated more easily. This happens with also using GoDraw, of course, but either too slowly or based on a certain “luck factor” be-

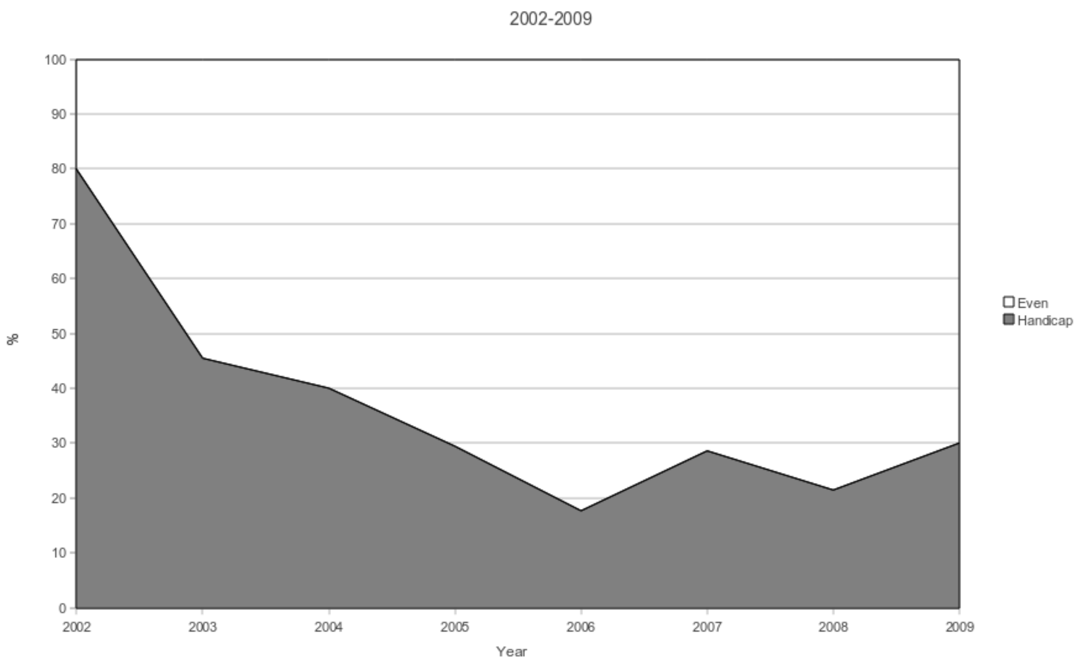


Figure 5.4: Tournament type percentages in Finland

cause of the randomizing. A normal weekend tournament of five rounds does not necessarily yield as much grading information as desired.

Chapter 6

EPM

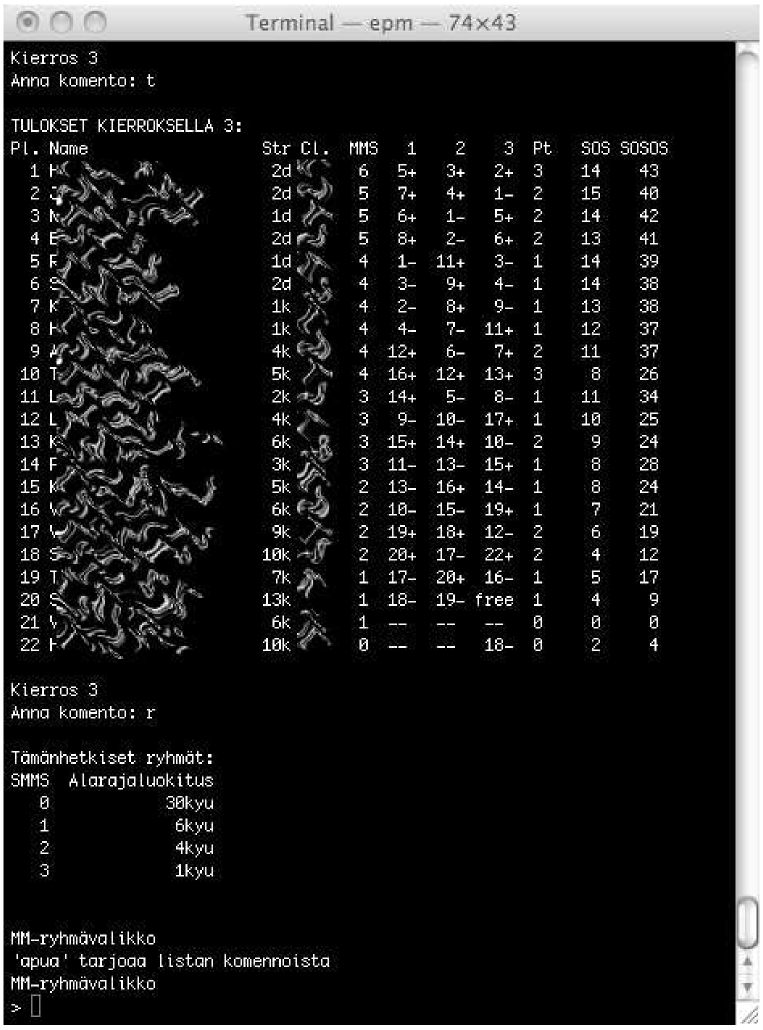
EPM, which stands for “Ein Pairung Maschin” is a tournament management software I designed because I found that other programs did not offer what I and a few other Finnish tournament managers needed.

EPM started as a round robin pairing program in 2003. When I could not think of any feature I could add, I decided to implement McMahon pairing. EPM also includes features for so called “bonus tournaments” where a player gets a point for a win and fractions of a point (usually 0.5 points) by consuming a standardized dose of ethanol.

EPM is implemented in C++ and currently supports only a text-based user interface which enables compiling and running it on all POSIX [9] compliant operating systems, for example Windows, Linux, and Mac OS X. A brief glimpse of the user interface can be seen in Figure 6.1.

6.1 Main principles

A few adjustments to calculating tournament standings have been made in order not to give anyone what they have not earned either by having a grade high enough or winning their games. Most of tournament for example give players half a point for every round they are absent (and usually round the absence points down). This is understandable for the player to “keep up” with everyone else while he is absent because all the other players receive on average half a point every round. This way the pairings are not affected by the absence.



```
Kierros 3
Anna komento: t

TULOKSET KIERROKSELLA 3:
Pl. Name                   Str Cl.  MMS   1    2    3  Pt    SOS SOSOS
  1 [redacted]              2d [redacted]  6   5+   3+   2+  3    14    43
  2 [redacted]              2d [redacted]  5   7+   4+   1-  2    15    40
  3 [redacted]              1d [redacted]  5   6+   1-   5+  2    14    42
  4 [redacted]              2d [redacted]  5   8+   2-   6+  2    13    41
  5 [redacted]              1d [redacted]  4   1-  11+   3-  1    14    39
  6 [redacted]              2d [redacted]  4   3-   9+   4-  1    14    38
  7 [redacted]              1k [redacted]  4   2-   8+   9-  1    13    38
  8 [redacted]              1k [redacted]  4   4-   7-  11+  1    12    37
  9 [redacted]              4k [redacted]  4  12+   6-   7+  2    11    37
 10 [redacted]              5k [redacted]  4  16+  12+  13+  3     8    26
 11 [redacted]              2k [redacted]  3  14+   5-   8-  1    11    34
 12 [redacted]              4k [redacted]  3   9-  10-  17+  1    10    25
 13 [redacted]              6k [redacted]  3  15+  14+  10-  2     9    24
 14 [redacted]              3k [redacted]  3  11-  13-  15+  1     8    28
 15 [redacted]              5k [redacted]  2  13-  16+  14-  1     8    24
 16 [redacted]              6k [redacted]  2  10-  15-  19+  1     7    21
 17 [redacted]              9k [redacted]  2  19+  18+  12-  2     6    19
 18 [redacted]             10k [redacted]  2  20+  17-  22+  2     4    12
 19 [redacted]              7k [redacted]  1  17-  20+  16-  1     5    17
 20 [redacted]             13k [redacted]  1  18-  19- free  1     4     9
 21 [redacted]              6k [redacted]  1  --   --   --   0     0     0
 22 [redacted]             10k [redacted]  0  --   --   18-  0     2     4

Kierros 3
Anna komento: r

Tämänhetkiset ryhmät:
SMMS  Alarajaluokitus
   0            30kyu
   1             6kyu
   2             4kyu
   3             1kyu


MM-ryhmävalikko
'apua' tarjoaa listan komennoista
MM-ryhmävalikko
> 
```

Figure 6.1: EPM running on Mac OS X, results view and McMahon group menu visible

However, if a player is absent for more than two rounds, the effect of the extra points can be seen in the players MMS points, SOS, and SOSOS (and therefore the tournament standings). This is not always desired. This is why EPM gives half a point for each round absent, rounds the total point sum down, and uses these points only when calculating pairings and the opponents' SOS.

When selecting players to be paired up or down, EPM selects players from the middle of the group instead of trying to even the players' SOS in a group.

Like Christoph Gerlach's MacMahon, EPM uses a weighting function to calculate weights for potential pairs. The pairing itself is not, however, done using a graph algorithm but a quite simple backtracking algorithm finding the best available pair for each player starting from the top, one group at a time. The players to be paired up or down are picked first. The backtracking enables the program to avoid locks which appear mostly on the bottom when the only two players left to pair have already played against each other.

The algorithm:

```
 1: Select the players to be paired up and down and pair them.
 2: A list, sorted by the weights, of preferred opponents is created for every player.
 3: Set the variable $Z$ for each player telling which opponent from the list should be exam-
    ined next to point to the first item.
 4: while unpaired players exist do
 5:    Pick the highest-standing player not paired.
 6:    Pick the first available opponent for this player from the list created earlier using the
       variable $Z$. Update $Z$ if needed.
 7:    If an opponent is found, store the pair. Continue from line 4.
 8:    If there are no available opponents for the player chosen on line 5, remove the previ-
       ous pair stored on line 7. Choose the higher standing player $A$ of this pair to be paired
       next. Reset $Z$ to point to the first item is the list of all players whose standing is worse
       than $A$'s.
 9: end while
10: Check the Z of the highest player in the standings. If it is past the end of his list created
    on line 2, the pairing could not be completed.
```

Chapter 7

Test runs

Though there is no absolute way to determine if one pairing is better than the other, some conclusions can be made by calculating player's relative strength (RS) and examining how close one's opponents' RS are to his. This is done by calculating the sum of RS differences squared. This way the total "error" of the pairing can be estimated because what one would consider as an optimal pairing consists of games between equally strong players.

For calculating the relative strengths for the players, I developed a very simple iterative algorithm where the results of the game either push the players' RSs further away from each other or draw them closer together depending on the RS difference and the result of the game. The RSs are calculated iteratively again and again until a satisfactory balance has been obtained. The algorithm is described in more detail in appendix B.

When the RSs have been calculated, the total error of the pairing is calculated by

$$e_{total} = \frac{1}{2kN}\sqrt{\sum_{n=0}^{N-1}\sum_{r=1}^{k}(RS(n) - RS(opponent(n,r)))^2}, \tag{7.1}$$

where N is the number of players, k the number of rounds, and $opponent(n, r)$ returns the player n's opponent on round r. $\frac{1}{2kN}$ removes the effect of differently sized tournaments and prevents each game being counted twice.

The function is derived from the statistical method of "the sum of squares", used for example in regression and defined by the sum of the squares of the difference of the dependent variable y and its grand mean $\bar{y}$: ($\sum_{n=0}^{N-1}(y_n - \bar{y})^2$). In this case we do not want to compare the relative strengths to their mean but to each other.

Table 7.1: Total errors of different tournaments, MacMahon

Tournament	Year	Players	Rounds	Total error
Stockholm Open	2004	43	5	0.1984
Toyota Pandanet European Go Tour Hamburg	2005	196	7	0.2754
European Go Congress	2005	712	10	0.3528
Leksand	2006	46	5	0.2110
1st Rabbity Six handicap (Hungary)	2007	32	5	0.3113
Hungarian GP	2007	68	5	0.2465
Turku GP	2007	34	5	0.1936

Table 7.2: Total errors of different tournaments, Gotha

Tournament	Year	Players	Rounds	Total error
Toyota Pandanet European Go Tour Paris	2005	329	6	0.4178
Toyota Pandanet European Go Tour Paris	2006	302	6	0.4074
Amiens Tournament	2007	61	4	0.3727
Levallois Tournament	2007	89	5	0.3155
Bale Tournament	2007	27	5	0.1427
Go Marathon	2007	67	8	0.1437

Of course this method does not only estimate the pairing, but also the way McMahon groups are made has a big effect. This effect diminishes when the number of players increases, groups become larger and there are less gaps in the grade distribution.

The results for these test runs can be found in tables 7.1 to 7.4. The tournaments were selected from the last few years trying to include tournaments of different sizes.

In a few cases, the largest errors are found at the largest tournaments. This is probably caused by larger tournaments having more players who do not play on all the rounds and receiving free points based on “probable success” which may not be correct. When they are paired according to these points, the pairings may be off. However, the differences between the errors of the largest tournaments and the next biggest errors are relatively quite small.

Table 7.3: Total errors of different tournaments, GoDraw

Tournament	**Year**	**Players**	**Rounds**	**Total error**
London Open Go Congress	2004	134	8	0.2289
British Go Congress	2007	98	6	0.2337
Scottish Open	2007	38	6	0.1638
Welsh Open	2007	37	5	0.1971
UK Go Challenge Finals	2007	51	6	0.3490
Durham	2007	38	6	0.1908
Epsom	2007	51	3	0.3278

Table 7.4: Total errors of different tournaments, EPM

Tournament	**Year**	**Players**	**Rounds**	**Total error**
Oulu Summer Tournament	2005	38	6	0.1705
Europen Students’ Go Championship	2005	36	6	0.1879
Yläkaupungin Yö	2006	40	5	0.2127
Oulu Summer Tournament	2006	32	6	0.1554
Toyota Tour Tampere	2006	91	5	0.2316
Takapotku Open	2007	69	5	0.2308

Chapter 8

Conclusion

The tournament management programs tested in this thesis do their job quite well. Problems with different programs seem to be almost the same, often related to the shortcomings of the user interface. The test runs show that the pairing algorithm used does not make a big difference but the data given to the algorithm, more promptly the way the weights for different pairs are determined, is the crucial part.

However, all the programs are not fit for use everywhere. GoDraw for example is quite unusable in Finland because of the need for grading information. With larger tournaments and with smoother grade distribution the differences between the programs diminish maybe due to averaging.

One big flaw can be found in most of the programs and it is the use of SOS (or any other secondary placement criteria) to make pairings. Using SOS to make pairings usually leads to equalizing SOS inside groups which pretty much renders SOS totally useless as a tie-breaker. Therefore a tie-breaker not so dependent on the pairings should correct the problem or at least make it more bearable but five rounds is far too little information for any system to achieve reliable results. Using seeding is fine as long as the pairing is not based on a placement criteria and at the same time try to minimize the differences of that same criteria between the players.

The same principles used in Finland to obtain more grading information could be used in other countries in Europe to make GoR work better. Because the main problem of GoR concerns players improving faster than the system expects it could adapt faster if given more optimized information.

For future research, statistical methods like the one described at 2.4.5 could be one possible direction to move to.

Bibliography

[1] *British Go Association Ratings FAQ,*
http://www.britgo.org/rating/krfaq.html, January 2007

[2] *Crazy Stone Wins Second UEC Cup*
http://remi.coulom.free.fr/CrazyStone/UEC2008/, May 2009

[3] Rémi Coulom, *Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search*
http://remi.coulom.free.fr/CG2006/CG2006.pdf, May 2010

[4] *Glossary of statistical terms,*
http://www.stat.berkeley.edu/ stark/SticiGui/Text/gloss.htm, January 2007

[5] Christoph Gerlach, *Ein MacMahon-Losungsprogramm für Go-Turniere unter Benutzung vom Maximum Weight Perfect Matching* 1994

[6] *Nederlandse Go Bond Homepage,*
http://gobond.nl/cgi-bin/Generiek/ToonPagina.cgi?Pagina=14&ID=&SID=&Target=All, January 2007

[7] Nihon ki-in (Japanese Go Association), *Nihon igo kiyaku*
Rules of Go (in Japanese), 1989

[8] Jack Edmonds, *Paths, trees, and flowers,*
Canadian Journal of Mathematics, 17:449-467 1965

[9] *POSIX - WikiPedia, the free encyclopedia,*
http://en.wikipedia.org/wiki/POSIX, February 2008

[10] *Sensei's Library,*
http://sensei.xmp.net/, June 2006.

[11] *SuomiGo-wiki*,
http://www.suomigo.net/, June 2006

[12] Nicol N. Schraudolph, Peter Dayan, Terrence J. Sejnowski, *Temporal Difference Learning of Position Evaluation in the Game of Go* 1994

Appendix A: Mathematical glossary

Go games can be handled as statistical phenomena and the whole problem of optimal pairing as an optimization of statistical measures based on information obtained from games. Statistics is a quite important tool when dealing with Go pairings and therefore a few terms should be explained.

expected value Given a discrete random variable *k(n) (n=0..N-1)* with a countable number of possible values and probabilities of each possible value of *k(n)* is given by *p(n)* *(*$\sum_{n=0}^{N-1} p(n) = 1$*)* the expected value of k is the weighted average of the possible values, $\sum_{n=0}^{N-1} p(n)k(n)$.

histogram A histogram is a plot that summarizes how data are distributed. The number of each possible value is counted and the plot is made with possible values on the X-axis and their numbers on the Y-axis. [4]

maximum likelihood estimate, MLE The maximum likelihood estimate of a parameter from data is the possible value of the parameter for which the chance of observing the data largest. That is, suppose that the parameter is p, and that we observe data x. Then the maximum likelihood estimate of p is estimate *p by the value q that makes P(observing x when the value of p is q)* as large as possible.

mean Often referred to as "average". The mean of *k* is calculated by $\frac{1}{N}\sum_{n=0}^{N-1} k(n)$.

nearly normal distribution A population of numbers has a *nearly normal distribution* if the *histogram* of its values nearly follows a *normal curve*.

normal curve Normal curve, also known as the "bell curve" is defined by $y = \frac{1}{\sqrt{2\pi e^{x^2}}}$.

normal distribution A random variable X has a normal distribution with *mean* m and *standard error* s if for every pair of numbers $a \leq b$, the chance that $a < \frac{X-m}{s} < b$ is $P(a < \frac{X-m}{s} < b)$ = area under the normal curve between a and b. If there are numbers m and s such that X has a normal distribution with mean m and standard error s, then X is said to have a normal distribution or to be normally distributed. If X has a normal distribution with mean $m = 0$ and standard error $s = 1$, then X is said to have a standard normal distribution.

population Population is a collection of units being studied. The units can be cars, people, Go players or just about anything.

probability The probability describes how likely a given event will happen. Probability is usually marked with $0 \leq P(event) \leq 1$. The sum of the probabilities of possible events is always 1.

random experiment An experiment or trial whose outcome is not perfectly predictable, but for which the long-run relative frequency of outcomes of different types in repeated trials is predictable.

random variable Assignment of numbers to possible outcomes of a *random experiment*.

standard error The Standard Error of a random variable is a measure of how far it is likely to be from its expected value; that is, its scatter in repeated experiments. The SE of a random variable X is defined to be $SE(X) = \sqrt{E((X - E(X))^2)}$. That is, the standard error is the square-root of the expected squared difference between the random variable and its *expected value*.

variable A numerical value or a characteristic that can differ individual members of a *population*.

Appendix B: Relative strength calculation

```
Gather the players' resultlines from the tournament results
for $n = 1$ to $N$ do
    $RS(n) \leftarrow 0$
end for
$totalerror \leftarrow 10 * N$ {repeat until the strength estimators are quite stable}
while totalerror > 0.001*N do
    $totalerror \leftarrow 0$
    for $n = 1$ to $N$ do
        $RS_{old}(n) \leftarrow RS(n)$
    end for
    for $n = 1$ to $N$ do {for each player...}
        for $r = 1$ to $k$ do {for each round...}
            $o \leftarrow opponent(n, r)$ {find the opponent for that round}
            sdiff $\leftarrow RS_{old}(n) - RS_{old}(o)$
            if $result = +$ then
                if sdiff $> 0$ then
                    $RS(n) + (1 - log_{\frac{N}{5}}(|\text{sdiff}| + 1)) - > RS(n)$,
                else
                    $RS(n) + log_{\frac{N}{5}}(|\text{sdiff}| + 1) - > RS(n)$
                end if
            end if
            if $result = -$ then
                if sdiff $> 0$ then
                    $RS(n) - log_{\frac{N}{5}}(|\,\mathrm{sdiff}\,| + 1) -> RS(n)$
                **else**
                    $RS(n) - (1 - log_{\frac{N}{5}}(|\,\mathrm{sdiff}\,| + 1)) -> RS(n)$
                **end if**
            **end if**
            **if** $result ==$ **then**
                **if** $\mathrm{sdiff} > 0$ **then**
                    $RS(n) - log_{\frac{N}{5}}(|\,\mathrm{sdiff}\,| + 1) -> RS(n)$
                **else**
                    $RS(n) + log_{\frac{N}{5}}(|\,\mathrm{sdiff}\,| + 1) -> RS(n)$
                **end if**
            **end if**
        **end for**
        Limit RS(n) to $0 \leq RS(n) \leq N - 1$
    **end for**
    Normalize the RSs so that that $0 \leq RS(n) \leq N - 1$, $0 \leq n \leq N - 1$.
    **for** $n = 1$ to $N$ **do**
        $totalerror \leftarrow totalerror + (RS(n) - RS_{old}(n))^2$
    **end for**
**end while**
```