Designing chess pairing mechanisms

Péter Biró[1]

Institute of Economics, Research Centre for Economic and Regional Studies, Hungarian Academy of Sciences, H-1112, Budaörsi út 45, Budapest, Hungary, and Department of Operations Research and Actuarial Sciences, Corvinus University of Budapest
peter.biro@krtk.mta.hu

Tamás Fleiner[2]

Department of Computer Science and Information Theory
Budapest University of Technology and Econimics
1117 Budapest, Magyar tudósok körútja 2., Hungary
fleiner@cs.bme.hu

Richárd Palincza

Department of Computer Science and Information Theory
Budapest University of Technology and Econimics
1117 Budapest, Magyar tudósok körútja 2., Hungary
richard.palincza@gmail.com

Abstract: The Swiss system is the most popular chess tournament system that is recognised and regulated by the World Chess Federation (FIDE). Chess pairings in each round of a Swiss tournament are conducted by sophisticated matching algorithms. The matching mechanisms are precisely defined in the FIDE guidebook [3], currently four different variants are allowed. The descriptions of the matching procedures are such that every arbiter should be able to conduct the pairings, even without computer assistance. However, many parts of these procedures are very inefficient, as they may terminate in highly exponential time in the number of players due to their exhaustive search nature. We demonstrate how the main priority rules of the Dutch variant can be replaced by efficient matching algorithms. These efficient algorithms can serve as the base of software tools used for pairings.

Keywords: maximum weight matching, mechanism design, roommates problem, tournaments

1 Introduction

The Swiss system is a pairing system invented by Dr. Julius Müller of Brugg, Switzerland. It was first used in a chess tournament at Zurich in 1895. It has been used in the Unites States since 1942 and also the team chess world championship, the so-called Olympiad was first organised with the Swiss system in Buenos Aires in 1978. The World Chess Federation (FIDE) allow currently four variants of the Swiss system to be used in individual tournaments, namely the Dutch system, Lim, Dubov and Burnstein systems. In this paper we focus on the Dutch system, which is the most classical one among the four.

[1]Research is supported by the Hungarian Academy of Sciences under its Momentum Programme (LP2016-3/2016), and by OTKA grant no. K108673.
[2]Research is supported by OTKA grant no. K108383.

The corresponding matching procedures are precisely described in the FIDE guidebook [3], with a new guidance published very recently on the Dutch system to be implied from 1 July 2017. We note that the latest version of the Dutch system is much clearer than the previous one, the requirements and the priorities on the aimed pairing are described in a mathematically more structured way. We include the most important part of the pairing rule in the Appendix.

Yet, these matching procedures are still rather complicated and their descriptions contain some exhaustive search routines which can make the pairing very slow, i.e. highly exponential in the number of players. In this research we investigate whether some of these routines can be replaced by efficient matching algorithms, e.g. by the maximum size and maximum/minimum weight matching algorithms of Edmonds. A similar investigation has been done by Ólafsson [5], but his study was concerned with an older and simpler version of the Swiss system. In particular, the transposition and exchanges rules were not considered in his work, which are two highly exponential routines in the current versions (both in the currently used rule and also in the new one valid from 1 July 2017).

Finally, let us note that even if the translation of the current rules to efficient algorithms is possible for some routines, it can be still reasonable to keep the exhaustive search descriptions in the official descriptions as some arbiters might still do the pairings by hand (and it would be too demanding for the arbiters if FIDE would request them to conduct the Edmonds' algorithm by hand). However, in most tournaments the pairing are conducted by some software, and in their pairing algorithm it would be indeed useful to replace the inefficient routines with efficient algorithms. By our mathematical investigation we also aim to understand the priorities used in the matching process which implies the properties of the matchings obtained, as these are not obvious from the current descriptions of the matching rules.

First we describe the basic notions, rules and goals of the pairings, and then we investigate the particular routines used in the Dutch variant.

2 Basic description of the Swiss pairings

In this section we summarise the basic features of the Swiss pairings, the concepts, definition, rules and common priorities used in all of the four variants.

2.1 The basic rules and goals

The general goal of a chess tournament is to select the winner and rank the others. The most important requirement of the pairing is that everyone should play in each round of the tournament (except one player if the number of players is odd). Thus, in mathematical terms, the matching obtained should be (almost) complete. In the case of odd number of players one player will remain unmatched in every round, and gets a bye (i.e. 1 point) without colour. This cannot happen twice with any player during a tournament.

The second common criterion in the Swiss tournament is that no pair of players can play twice. Thus when considering a pairing problem these pairs are not eligible.

Finally, since playing a game with white or black can have significant effect on the result, a tournament is considered fair if every player has played approximately the same number of times with white and black. These colour rules are a bit softer and used differently in the four variants, but a common strict rule is not to let any player to play with the same colour three times in a row, and also not to let any player to have a colour difference greater than two. There are however some exceptions with regard to these rules in the very last round of a tournament.

2.2 Implementation of the goals

The basic concept of the Swiss pairing is to rank the players after each tournament according to their scores, and to match them from the strongest ones to the weakest ones sequentially according to their scores. To understand the procedure, first we have to describe the scoring rules of an individual chess tournament for the general readership. In a chess tournament with individual players a player gets score

1 if she wins, score half is she draws and zero score if she looses. A typical Swiss tournament has nine rounds, and after that the players are primarily ranked according to their total scores. The pairing procedure considers the players according to their scores and tries to match everyone to someone with the same score, starting with the strongest players. Thus after, say, five rounds of a Swiss tournaments, the process is to consider those players first those who have 5 points, and then of those with 4.5 points, and so on. The pairing process is described for players within each score-group.

Sometimes it is not possible to match everyone within the score group, as not all the pairs are eligible and also because we may have an odd number of players. In this case the task is to match as many players as possible, and the rest will be moved to the subsequent score group. These players are called *downfloaters*. In order to avoid the further downgrade of downfloaters when considering the subsequent score-group, the downfloaters must be all matched, if possible. Their opponents will be called *upfloaters*. As a weak rule, we may also want to avoid to select the same players to become down- or upfloaters, so we shall try not to give an identical float to any player in two consecutive rounds or twice in three consecutive rounds.

2.3 Mathematical notions

We describe the pairing problem of a chess tournament as a matching problem in a nonbipartite graph $G(N, E)$ with node set $N = \{1, 2, \dots n\}$ and edges set $E = \{e_1, e_2, \dots, e_m\}$. The players correspond to nodes and we have an edge $ij \in E(G)$ if players i and j are eligible to play with each other. A *matching* is a set of independent edges in E, i.e. every node is incident with at most edge in a matching. An *(almost) complete* matching has size $\lfloor \frac{n}{2} \rfloor$ in G.

Finding a maximum size or maximum weight matching in a nonbipartite graph can be done efficiently by Edmonds' algorithm. The best implementation for the maximum size algorithm has running time $O(\sqrt{n} \cdot m)$ according to [6], and the best currently known running time for the maximum weight matching algorithm is $O(nm + n^2 \log n)$ due to [2].

The selection of the pairing is based on various priority rules. As we will demonstrate, finding the right pairings can be done by using exponentially decreasing weights, or equivalently to do the optimisation with *weight-vectors* on the edges. We choose the latter technique to make the description simpler. In particular, for each edge $e = ij$ we will introduce a weight vector w_e of $O(n)$ length. The implementation of the pairing rule will be equivalent to finding a matching on the graph with a lexicographically maximal weight. Note that using the weight-vectors will increase the running time of the classical Edmonds algorithm by a factor of n, but still remains strongly polynomial in the number of players (n).

3 The Dutch system

The official description of the currently used Dutch system describes the subroutine used in the most inner cycle of the pairing process, and the exhaustive search method are then extended for the case when no ideal pairing is possible. However, in the new description (to be applied from 1 July 2017) these subroutines are only suggested to use after satisfying the main criteria and goals. Thus we will also follow the new description (partly included in the Appendix) and first describe the main criteria and then the transposition and exchange rules. After providing the short description we show how the subroutines can be implemented with efficient matching algorithms.

3.1 Summary of the FIDE description

Suppose that we consider a score-group S during the pairing process. If this is not the highest score-group then there may be some downfloaters, denoted by F. First we divide S into two subgroups $S1$ and $S2$ of approximately the same size, where $F \subseteq S1$, and $|S1| \leq |S2| \leq |S_1| + 1$, if possible. In the running example of the official guide we have $S = \{1, 2, \dots, 11\}$ and $S1 = \{1, 2, \dots, 5\}$, $S2 = \{6, 7, \dots, 11\}$.

Eligibility criteria

As described in points C1-C3 in the Appendix, some players are not eligible to be paired. Essentially no two players can be matched twice, no player can become unmatched and thus get a bye twice, and the absolute color preference of a non-topscorer player must be obeyed, so we can never match two non-topscorer players with the same absolute color preference.

Priorities for selecting the pairing

The new version of the FIDE Dutch system rules includes a clear prioritisation order over the pairing selected, which is included in the Appendix.

First, we have to note that slightly different rules are applied for score groups at the end of the process. The so-called completion criteria C4 requires that in the score group before the last one we shall choose the downfloaters in such a way that the last group will admit a complete matching.

The further criteria (C5-C19) are called quality criteria, and indeed these provide the sequential goals that the best pairing should satisfy, see them in the Appendix. The first criterion (C5) is to maximise the size of the pairing within the score group considered. The second criterion (C6) is to maximise the number of downfloaters paired, and among them match the ones with the highest scores. The following rule C7, was not present in the previous version of the Dutch rule, and it is a forward looking rule that requires the selection of the downfloaters in the considered score group such that in the subsequent score group the pairing has maximum size and matches the most downfloaters. The remaining rules C8-C19 provides a specific order how the colour preferences and the repetition of the downfloter and upfloater selection are considered. (Note that here do not consider the exceptions that apply for the very last score group, the first and the last rounds of the tournament, or in case of some other unusual events, e.g. withdrawal or addition of players, unfinished games.)

Tie-breaking by transpositions and exchanges

When the above described priority rules do not provide a unique solution (which is typically the case at the beginning of the tournaments, where the score groups are large and the eligibility and priority criteria are easier to satisfy) the rule suggest a particular order among the possible matchings. The transposition order describes the rankings of the matchings when the set of players to be matched, $S1$ and $S2$, are already fixed. The exchange rules describe in which order one shall try to exchange the players among $S1$ and $S2$ in when the satisfaction level of the priority criteria is already fixed. Thus, in fact the exchange order is more important and we should consider that first, but below we follow the description of the FIDE Handbook and we start describing the transposition orders.

Transposition orders. The most inner process of the Dutch pairing algorithm will select the first feasible pairing between $S1$ and $S2$ as follows. The pairings are sorted according to a lexicographic order considering the first player in $S1$ first, then the second player in $S1$, and so on. For our running example, the players in $S1$ in their order shall get the following opponents, the first suitable pairing from the list described in Table 1.

Exchange orders. If neither of these pairings is eligible then we need to try to exchange players between sets $S1$ and $S2$ in a predefined order, as described in part D2-D3 in the Appendix. For instance, if we can find a suitable pairing by exchanging one pair of players then we should check the pairs to be exchanged in the order described in Table 2.

If more than one pair of players are needed to be exchanged then the rule requires to

1. minimise the number of players exchanged
2. minimise the index differences between the players exchanged
3. lexicographically maximise the indices of the players moved from $S1$ to $S2$

0.	6-7-8-9-10-11
1.	6-7-8-9-11-10
2.	6-7-8-10-11-9
3.	6-7-8-11-9-10
4.	6-7-8-11-9-10
5.	6-7-9-8-10-11
...	...
12.	6-7-10-8-9-11
...	...
24.	6-8-7-9-10-11
...	...
719.	11-10-9-8-7-6

Table 1: Transposition order when pairing $S1 = \{1, 2, 3, 4, 5\}$ and $S2 = \{6, 7, 8, 9, 10, 11\}$.

	5	**4**	**3**	**2**	**1**
6	1	3	6	10	15
7	2	5	9	14	20
8	4	8	13	19	24
9	7	12	18	23	27
10	11	17	22	26	29
11	16	21	25	28	30

Table 2: Priority order when exchanging one pair of players between $S1 = \{1, 2, 3, 4, 5\}$ and $S2 = \{6, 7, 8, 9, 10, 11\}$.

4. lexicographically minimise the indices of the players moved from $S2$ to $S1$

For instance, when considering the exchange of two players from each group, we shall use the following priority order described in Table 3.

3.2 Translating the rules into efficient algorithms

In this section we describe how to translate the selection rules into a maximum weight matching algorithm. Note that here, we focus on the regular cases.

Eligibility requirements. The eligibility requirements (C1-C3) can be easily satisfied by not having edges between the nodes representing these agents in the eligibility graph $G_S(N, E)$ for score group S.

Completion criterion. The completion criterion (C4) is only applied for the score group that is considered before the last one, and as a first priority we have to make it sure that the last score group will have a complete matching. So essentially we have to find a complete matching for the last two score groups. (It is not mentioned in the latest version of the rule what would happen if there exist no complete matching for the last two groups, but in the earlier version they recommend to enlarge the set of players considered with the previous score group(s).)

Quality criteria. Each of the quality criteria (C5-C17) can be translated into a maximum weight matching problem with weight-vectors. For each selection criterion we define a new index for the weight-vector w_e for every edge $e = ij$ as follows.

1. For (C5) we simply set weight 1 for each edge. Maximising this index will ensure that the matching

	5,4	**5,3**	**5,2**	**5,1**	**4,3**	**4,2**	**4,1**	**3,2**	**3,1**	**2,1**
6,7	1	3	7	14	8	16	28	29	45	65
6,8	2	6	13	24	15	27	43	44	64	85
6,9	4	11	22	37	25	41	60	62	83	104
6,10	9	20	35	53	39	58	79	81	102	120
6,11	17	32	50	71	55	76	96	99	117	132
7,8	5	12	23	38	26	42	61	63	84	105
7,9	10	21	36	54	40	59	80	82	103	121
7,10	18	33	51	72	56	77	97	100	118	133
7,11	30	48	69	90	74	94	113	115	130	141
8,9	19	34	52	73	57	78	98	101	119	134
8,10,	31	49	70	91	75	95	114	116	131	142
8,11	46	67	88	108	92	111	126	128	139	146
9,10	47	68	89	109	93	112	127	129	140	147
9,11	66	87	107	123	110	125	137	138	145	149
10,11	86	106	122	135	124	136	143	144	148	150

Table 3: Priority order when exchanging two pairs of players between $S1 = \{1, 2, 3, 4, 5\}$ and $S2 = \{6, 7, 8, 9, 10, 11\}$, as given in the FIDE Handbook.

in maximum size.

2. For (C6) we set weight 1 if either i or j is a downfloater, and 0 otherwise. (Note that there cannot be an edge between two downfloaters, since in that we would had matched them before). This weighting will ensure that we match as many downfloaters as possible.

3. Still corresponding to (C6), we need to match first those downfoaters who have the highest scores and continue with the second highest ones. This can be achieved by adding a weight-vector for every eligible pair containing a downfloater, which is a zero-one vector as long, as the number of different scores of the downfloaters. For instance, if the score group considered contains players with score 4 and there are downfloaters with scores 5.5 and 4.5 then we add a vector of length two, and first we put a value 1 to those players with score 5.5 and then a value 1 for those with score 4.5, leaving the other values zero.

4. Rule (C7) is a special one, as we will need to ensure the maximality of the matching in the subsequent score group, denoted by S', and also the number of downfloaters matched there. For this rule, we extend graph G_S to graph $G_{S \cup S'}$, and we only define weights with regard to this index for edges between S and S' and within S'. Let the weight of these edges be 1, ensuring first the maximality of the matching in the subsequent score group.

5. To ensure that the number of downfloaters matched is also maximal when matching the subsequent group S', according to (C7), we add weight 1 of each edge between S and S'.

6. Rules (C8)-(C9) only apply for the topscorers and their opponents in the last round (i.e. "players who have a score of over 50% of the maximum possible score when pairing the final round of the tournament"), as a relaxation of eligibility criteria (C3). In case we are considering these players we add weight -1 for those edges where both players have the same absolute colour preference, first by the fact that either they both have +2 or -2 colour difference.

7. Continuing the above rule by part (C9), we also add weight -1 for those pairs who both played with the same colour two times in a row.

8. To ensure that most player get their colour preferences according to (C10), we add weight -1 if both players have the same colour preference.

9. Similarly, we can minimise the number of player who do not get their strong colour preference, as required in (C11), by adding weight -1 if both players have the same strong colour preference.

10. To satisfy (C12), if either of the two players involved was a downfloater in the previous round then we add weight 1, so the algorithm will try to match as many of them as possible, and avoid to select them to become downfloaters again. (+1 if one player, +2 if both)

11. Minimising the selections of the same players for becoming upfloaters, as described in (C13), we add weight -1 for edge ij if i is a current downfloater (i.e. i was unmatched in the previous score group), and j was an upfloater in the previous round.

12. Selection rule (C14) can be treated in the same way as rule (C12). (+1/+2 as C12)

13. Selection rule (C15) can be treated in the same way as rule (C13).

14. In rule (C16) we need to minimise the score differences for the players who receive the same downfloat as in the previous round. So, if for pair ij, $i \in S$ was a downfloater in the previous round and $j \in S'$ then we add $-k$ as the weight if k is the difference between the scores of player i and j.

15. Similarly, in rule (C17) we minimise the score differences from the point of view of the repeated upfloaters, by adding weight $-k$ if the difference between the scores of downfloater i and previous upfloater j is k.

16. Rule (C18) can be treated as rule (C16).

17. Rule (C19) can be treated as rule (C17).

Finding a lexicographically weight-maximal matching with the above weighting on $G_{S \cup S'}$ will provide us a matching that we would select when sequentially maximising criteria (C5-C19). After optimising with regard to the quality criteria, we need to choose the pairing according to the transposition and exchange rules. Since the exchange rules are superior, we start the translation with that.

Exchange rule. We extend the above weight-vectors with the following components, responsible for enforcing the exchange selection. Here we describe the translation for so-called homogenous score groups (where no downfloaters are present), but the heterogenous case can be treated similarly.

1. To minimise the number of players exchanged we add weights -1 for every edge within $S1$ and within $S2$. Our optimal matching will use as few edges as possible, which also means that the number of players exchanged is minimal.

2. To minimise the index differences between the players exchanged we add the following negative weights. Let r_i be the index of player i, and let a_S denote the index between the highest index in $S1$ and the lowest index in $S2$ (this is 5.5 in our running example). For every edge ij, where $i, j \in S1$ and $r_i < r_j$ let the weight of ij in the vector be $r_j - a_S$. Similarly, for every edge ij, where $i, j \in S2$ and $r_i < r_j$ let the weight of ij in the vector be $a_S - r_i$. E.g. for edge $\{2, 4\}$ in our running example this weight is -1.5 and for edge $\{8, 9\}$ this weight is -2.5.

3. To lexicographically maximise the indices of the players moved from $S1$ to $S2$, we add a weight-vector of length $|S1|$ and with one nonzero element, as follows. If $i, j \in S1$ with $r_i < r_j$ then we add a weight 1 to the $\lceil a_S - r_j \rceil$-th coordinate. For instance, in our running example when considering edge $\{2, 4\}$, the added weight-vector is $[0, 1, 0, 0, 0]$. This weighting will ensure that we will move player 5 to $S1$ whenever it is possible, and if not then player 4, and so on.

4. To lexicographically minimise the indices of the players moved from $S2$ to $S1$ we further extend the weight-vector with a new component of length $|S2|$ and with one nonzero element. If $i, j \in S2$ with $r_i < r_j$ then we add a weight 1 to the $\lceil r_i - a_S \rceil$-th coordinate. For instance, when considering edge $\{8, 9\}$ in the running example, the added vector is $[0, 0, 1, 0, 0, 0]$. Thus we move player 6 first, if possible, then player 7, and so on.

Finally, for choosing the first pairing among the so far optimal ones, we translate the selection according to the transposition order into a maximum weight matching problem.

Transposition rule. With the transposition rule, we assume that the partition $S1 \cup S2$ is already fixed and we would like to ensure that the among the possible pairings we first select the partner of the player with the smallest index in $S1$ to be the player with the smallest index in $S2$, and if this is not possible then the player with the second smallest index in $S2$, and so on. After selecting the partner of the highest ranked player in $S1$, we continue with selecting a partner for the second highest ranked player in $S1$, and so on. To achieve this, we add another weight-vector component to each edge of length $|S| - 1$ as follows. For ij, where $r_i < r_j$ we add weight $-r_j$ on the r_i-th position in this vector and we keep the other position zero-valued. For instance, for edge $\{2, 7\}$ in our example, we add vector $[0, -7, 0, 0, 0, 0, 0, 0, 0, 0]$.

4 Further notes

In this paper we discussed how to replace the priority rules in the Swiss pairing systems by efficient algorithms. However, we have only studied the Dutch variant, and we have not considered some special cases (last round, heterogenous score groups, new or leaving players, etc). Nevertheless, we believe that all of the variants of the Swiss pairings can be completely conducted by efficient algorithms, so our first future plan is to investigate the remaining details of the Dutch rule and the three other systems.

If we succeed to translate the official pairing procedures into sophisticated efficient algorithms then we can incorporate them into a software and conduct further studies. In particular, it would be interesting to simulate tournaments and compare the performance of the four variants with respect to their success of ranking the players according to their real strength within the same number of rounds. This would follow up the research of Csató [1], who compared the final rankings of some particular tournaments organised by Swiss pairings with other ranking methods.

In a future research one could also investigate the performance of these variants from a more general point of view, by considering the utilities of the players. A player in a Swiss tournament may not be really interested in her final ranking, and the ranking of the others, as perhaps she just wishes to play with opponents of similar strength. Note that such preferences are not likely to be satisfied in the most widely used Dutch variant if a tournament has many participants, since according to the exhaustive search procedure (dividing a score group based to the ELO point of the players and then trying to match them according to their order in their subgroups), a typical player will either play with much stronger of with much weaker players in the first 5-6 rounds of the 9-round tournament. One alternative pairing method would consider the preferences of the players and match them e.g. with a stable matching algorithm, as proposed in [4]. Thus the four variants could be compared with respect to such preferences, namely how large is the gap between the strengths of the paired players in average during a tournament. Finally, it would also be interesting to see what kind of alternative pairings could be used to better satisfy the preferences of the players when the classical goal of selecting a winner and ranking the others is ignored.

References

[1] L. Csató. Ranking in Swiss system chess team tournaments. *Corvinus Economics Working Papers (CEWP) 2015/01, Corvinus University of Budapest*, 2015.

[2] H.N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. *In proceedings of SODA-1990: The first annual ACM-SIAM symposium on Discrete algorithms*, 1990.

Gabow, Harold N. "Data structures for weighted matching and nearest common ancestors with linking." Society for Industrial and Applied Mathematics, 1990.

[3] World Chess Federation (FIDE) Website. www.fide.com/fide/handbook. *Accessed on 14 February 2016.*

[4] E. Kujansuu, T. Lindberg, E. Mäkinen. The Stable Roommates Problem and Chess Tournament Pairings. *Divulgaciones Matemáticas*, 7(1):19–28, 1999.

[5] S. Ólafsson. Weighted Matching in Chess Tournaments. *The Journal of the Operational Research Society*, 41(1):17–24, 1990.

[6] S. Micali, V.V. Vazirani. An $O(\sqrt{|V|} \cdot |E|)$ algorithm for finding maximum matching in general graphs. *In: Proceedings of FOCS 1980: the 21st Annual Symposium on Foundations of Computer Science*, 17–27, 1980.

Appendix

FIDE Handbook, the Dutch system (to be applied from 1 July 2017)

C Pairing Criteria

Absolute Criteria

No pairing shall violate the following absolute criteria:
C.1 see C.04.1.b (Two players shall not play against each other more than once)
C.2 see C.04.1.d (A player who has already received a pairing-allocated bye, or has already scored a (forfeit) win due to an opponent not appearing in time, shall not receive the pairing-allocated bye).
C.3 non-topscorers (see A.7) with the same absolute colour preference (see A6.a) shall not meet (see C.04.1.f and C.04.1.g).

Completion Criterion
C.4 if the current bracket is the PPB (see A.9): choose the set of downfloaters in order to complete the roundpairing.

Quality Criteria
To obtain the best possible pairing for a bracket, comply as much as possible with the following criteria, given in descending priority:
C.5 maximize the number of pairs (equivalent to: minimize the number of downfloaters).
C.6 minimize the PSD (This basically means: maximize the number of paired MDP(s); and, as far as possible, pair the ones with the highest scores).
C.7 if the current bracket is neither the PPB nor the CLB (see A.9): choose the set of downfloaters in order first to maximize the number of pairs and then to minimize the PSD (see C.5 and C.6) in the following bracket (just in the following bracket).
C.8 minimize the number of topscorers or topscorers' opponents who get a colour difference higher than +2 or lower than -2.
C.9 minimize the number of topscorers or topscorers' opponents who get the same colour three times in a row.
C.10 minimize the number of players who do not get their colour preference.
C.11 minimize the number of players who do not get their strong colour preference.
C.12 minimize the number of players who receive the same downfloat as the previous round.
C.13 minimize the number of players who receive the same upfloat as the previous round.
C.14 minimize the number of players who receive the same downfloat as two rounds before.
C.15 minimize the number of players who receive the same upfloat as two rounds before.

C.16 minimize the score differences of players who receive the same downfloat as the previous round.
C.17 minimize the score differences of players who receive the same upfloat as the previous round.
C.18 minimize the score differences of players who receive the same downfloat as two rounds before.
C.19 minimize the score differences of players who receive the same upfloat as two rounds before.

D Rules for the sequential generation of the pairings

Before any transposition or exchange take place, all players in the bracket shall be tagged with consecutive in-bracket sequence-numbers (BSN for short) representing their respective ranking order (according to A.2) in the bracket (i.e. 1, 2, 3, 4, ...).

D.1 Transpositions in $S2$

A transposition is a change in the order of the BSNs (all representing resident players) in $S2$.
All the possible transpositions are sorted depending on the lexicographic value of their first N1 BSN(s), where N1 is the number of BSN(s) in $S1$ (the remaining BSN(s) of $S2$ are ignored in this context, because they represent players bound to constitute the remainder in case of a heterogeneous bracket; or bound to downfloat in case of a homogeneous bracket - e.g. in a 11-player homogeneous bracket, it is 6-7-8-9-10, 6-7-8-9-11, 6-7-8-10-11, ..., 6-11-10-9-8, 7-6-8-9-10, ..., 11-10-9-8-7 (720 transpositions); if the bracket is heterogeneous with two MDPs, it is: 3-4, 3-5, 3-6, ..., 3-11, 4-3, 4-5, ..., 11-10 (72 transpositions)).

D.2 Exchanges in homogeneous brackets or remainders (original $S1 \iff$ original $S2$)

An exchange in a homogeneous brackets (also called a resident-exchange) is a swap of two equally sized groups of BSN(s) (all representing resident players) between the original $S1$ and the original $S2$. In order to sort all the possible resident-exchanges, apply the following comparison rules between two resident-exchanges in the specified order (i.e. if a rule does not discriminate between two exchanges, move to the next one).
The priority goes to the exchange having:
a) the smallest number of exchanged BSN(s) (e.g exchanging just one BSN is better than exchanging two of them).
b) the smallest difference between the sum of the BSN(s) moved from the original $S2$ to $S1$ and the sum of the BSN(s) moved from the original $S1$ to $S2$ (e.g. in a bracket containing eleven players, exchanging 6 with 4 is better than exchanging 8 with 5; similarly exchanging 8+6 with 4+3 is better than exchanging 9+8 with 5+4; and so on).
c) the highest different BSN among those moved from the original $S1$ to $S2$ (e.g. moving 5 from $S1$ to S2 is better than moving 4; similarly, 5-2 is better than 4-3; 5-4-1 is better than 5-3-2; and so on).
d) the lowest different BSN among those moved from the original $S2$ to $S1$ (e.g. moving 6 from $S2$ to S1 is better than moving 7; similarly, 6-9 is better than 7-8; 6-7-10 is better than 6-8-9; and so on).

D.3 Exchanges in heterogeneous brackets (original $S1 \iff$ original Limbo) An exchange in a heterogeneous bracket (also called a MDP-exchange) is a swap of two equally sized groups of BSN(s) (all representing MDP(s)) between the original $S1$ and the original Limbo. In order to sort all the possible MDP-exchanges, apply the following comparison rules between two MDP-exchanges in the specified order (i.e. if a rule does not discriminate between two exchanges, move to the next one) to the players that are in the new $S1$ after the exchange.
The priority goes to the exchange that yields a $S1$ having:
a) the highest different score among the players represented by their BSN (this comes automatically in complying with the C.6 criterion, which says to minimize the PSD of a bracket).
b) the lowest lexicographic value of the BSN(s) (sorted in ascending order).
Any time a sorting has been established, any application of the corresponding D.1, D.2 or D.3 rule, will pick the next element in the sorting order.