DE GRUYTER OPEN
Acta Univ. Sapientiae, Informatica **9,** 1 (2017) 17–34

DOI: 10.1515/ausi-2017-0002

On partial sorting in restricted rounds

Dedicated to the memory of Antal Iványi

Antal Iványi
Eötvös Loránd University,
Faculty of Informatics
email:

Norbert Fogarasi
Budapest University of Technology and Economics
Department of Networked Systems and Services
email: fogarasi@hit.bme.hu

Abstract. Let n and k be integers such that $n \geq 2$ and $1 \leq k \leq n$. In this paper, we consider the problem of finding an ordered list of the k best players out of n participants by organizing a tournament of rounds of pairwise matches (comparisons). Assuming that (i) in each match there is a winner (no ties) (ii) the relative strength of the players is constant throughout the tournament and (iii) the players' strengths are transitive, the problem is equivalent to partially sorting n different, comparable objects, allowing parallelization in rounds. The rounds are restricted as one player can only play one match in each round. We propose concrete pairing algorithms and make conjectures about their performance in terms of the worst case number of rounds and matches required. The research article was started by professor Antal Iványi who sadly passed away during the work and was completed in his honor by the co-author. He hopes, in this modest way, to reflect his deep admiration for professor Iványi's many contributions to the theory, practice and appreciation of algorithm design and analysis.

Computing Classification System 1998: G.2.2.
Mathematics Subject Classification 2010: 05C85, 68R10
Key words and phrases: partial sorting, parallel sorting, tournament, pairing algorithm

1 Introduction

Sorting, that is ranking of objects using pairwise comparisons, is a common practical problem with application in different fields. Many authors describe different applications, e.g., Landau [51] biological, Hakimi [33] chemical, Kim et al. [46] and Newman et al. [57] network modelling, Bozóki, Csató, Fülöp, Kéri, Poesz, Rónyai and Temesi economical [6, 13, 14, 18, 19, 45], Liljeros et al. human relation modelling [52], while Csató, Iványi, Lucz, Móri, Pirzada, Reid, and Sótér [18, 30, 37, 38, 39, 41, 40, 43, 44, 62, 64, 65, 69] sport applications.

Partial sorting is a relaxed variant of the sorting problem in which the task is to return a list of the k largest (or k smallest) elements in order. A common practical example of partial sorting is computing the "Top 100" of some list. Martinez [56] has optimized the Quicksort algorithm for partial sorting. After unsuccessful attempts by Schreier [67] and Slupecki [68], in 1964 Kislitsyn [49] determined the number of necessary comparisons for $\mathsf{k} = 2$. Aigner [1] has proposed a general solution for $\mathsf{k} = 3$ but this has been improved by Eusterbrock [24] and Kirkpatrick [47, 48] and upper bounds have been established and proven for the required number of comparisons for $\mathsf{k} = 3$ by Hadian and Sodel [32] and Kirkpatrick [48].

Related problems are *unordered partial sorting* that is choosing the k largest elements in any order and *selection* which is choosing the k^{th} largest element of a given list. The selection problem has been extensively studied. There are many results on the lower and upper bounds of the number of necessary comparisons [10, 47, 71], near-optimal algorithms such as Ford-Johnson [28] and its improvements [7, 15, 53, 54, 55], and brute-force results [59, 60]. Finding both the largest and smallest elements at the same time has also been studied extensively [2, 3, 63, 70]. Of particular importance is the selection of the median element, for example in order to apply it as a pivot strategy for Quicksort. [22, 66, 21, 66]. The minimal and average number of necessary comparisons have been examined for median selection [32, 72, 20]. Knuth [50] remains an excellent survey of these problems and results.

With the advent of parallel computing devices, a natural direction for research is the parallelisation of sorting algorithms. Following the treatment of Pippenger [61], there are two different modes of parallelism for these problems: the case of a number of parallel comparisons equal to the number of elements which is called *balanced case*; and the case of a number of parallel comparisons large enough to allow the solution to be found in a fixed number of *rounds*: asking a fixed number of questions in the first round, processing the information then repeating this for a fixed number of rounds. This is called the *highly*

parallel case. For sorting n elements, it has long been known that, in the non-parallel case, $\Theta(n \log n)$ steps are needed. This implies that $\Theta(\log n)$ steps are needed in the balanced case, but Ajtai et al [4] showed that $\mathcal{O}(\log n)$ steps are sufficient. For the highly parallel case, Haggkvist and Hell [34] showed that $\Omega(n^{1+\frac{1}{k}})$ comparisons are needed to sort in k rounds which Alon et al [5] improved to a tighter bound. Bollobás and Thomason [11] showed that $\mathcal{O}(n^{\frac{3}{2}\log n})$ comparisons are sufficient to sort in 2 rounds which was improved by Alon et al. [5] and generalized by Bollobás and Hell [12] to $\mathcal{O}(n^{1+\frac{1}{k}\log n})$ comparisons for k rounds.

In this paper, we study the problem of partially sorting n distinct elements, in order to find the top k, in rounds, where only one pairwise comparison of each element to another is allowed in each round. We will call this a *restricted round* and is similar to sport tournaments where in each round, each team or individual can only play against one opposing team or individual. If we use the analogy of sports, we need to make the following three assumptions:

(i) in each match there is a winner (no ties)

(ii) the relative strengths of the players is constant throughout the tournament and

(iii) the players' strengths are transitive (i.e., if A beats B and B beats C then A beats C)

The task then is to come up with an optimal *pairing algorithm* which guarantees that the top k players can be found in a fixed number of rounds. Aziz et al. [8] study the related problem of determining possible and necessary winners for partially completed tournaments, Beasley et al. [9] also consider different ways of extending partial tournaments.

The structure of the paper is as follows: In section 2, formal definitions and notation are introduced, whilst in section 3 we review existing algorithms and present some related problems. In section 4, we propose some new algorithms and present some results, whilst in section 5 we draw conclusions and present directions for future research.

2 Definitions and notation

Throughout the paper, we will use the terminology of sports: the compared objects are called *players*, the comparisons *matches* and the comparisons in

each round the *pairing* of *round* i and corresponding ordering the *results* of round i. Let $n \geq 2$ be an integer denoting the total number of players and $1 \leq k \leq n$ the desired number of top players the pairing algorithm should find. In each match, the winner gets 1 point and the loser 0, so the set of permitted results is $\mathcal{R} = \{0:1, 1:0\}$. At the beginning of a tournament, each player is assigned an *index* which is an integer label $i \in \{1, \ldots, n\}$, while the *rank* of a player is the position of the player in the final ordering once the required comparisons have been made, which is equivalent to the value of the integer if we consider the equivalent problem of sorting integers $\{1, \ldots, n\}$ by pairwise comparisons.

A natural tool for the representation of the results of tournaments with n players is a directed graph G on n vertices (T_1, ..., T_n) or an $n \times n$ sized *point matrix* $\mathcal{M}$, where if player i gets $x \in \{0, 1\}$ points against player j then G contains x edges directed from T_i to T_j and $\mathcal{M}_{ij} = x$. As such, the results of tournaments can be represented by loopless directed graphs. Let $\Pi = \{\Pi_1, \ldots, \Pi_{n!}\}$ be the set of $n!$ possible permutations of the n players and $R_A(n, k, \Pi_i)$ denote the minimum required number of rounds needed for a given deterministic pairing algorithm A to rank the top k players out of n, under permutation Π_i. Our task is to find the algorithm with the least number of required rounds in the worst case, over all player permutations and the necessary number of rounds and games, i.e.,

$$R(n, k) := \min_{A \in \mathcal{A}} \max_{i} R_A(n, k, \Pi_i)\,, \tag{1}$$

where $\mathcal{A}$ is the class of all deterministic pairing algorithms which perform pairwise comparisons in rounds, pairing each player at most once in each round.

3 Existing algorithms and related problems

The two most commonly used tournament formats in sport tournaments are *round-robin* (all-play-all) and *knock-out* (elimination). Round-robin tournaments provide an upper bound for determining the full ordering, i.e., $R(n, 1) \leq R(n, 2) \leq \ldots \leq R(n, n) \leq n - 1$. Knock-out tournaments are efficient in determining the strongest player and illustrate the result that $R(n, 1) = \lceil \log_2 n \rceil$ [50]. Observing that the second best player must have been knocked out by the winner directly in one of the $\log_2 n$ rounds yields the result that $R(n, 2) = \lceil \log_2 n \rceil + \lceil \log_2 \lceil \log_2 n \rceil \rceil$. $R(n, 3)$ was first investigated by Carroll [16] who argued against the knock-out system for giving out multiple prizes

and proposed a novel algorithm to find the top three in a lawn tournament of 32 players.

A pairing algorithm which is widely used for chess tournaments with many participants (e.g., Chess Olympiad, large Open tournaments) is the *Swiss pairing system* (see [25, 26, 27]). This was first used in 1903 in the Swiss national tournament [58] although there are claims [35] that it was used in Zurich as early as in 1895. It was first formalized as an algorithm and programmed by Olafsson [58]. By construction, the Swiss pairing system has three main goals:

(i) Minimize the difference in the score of players paired against each other.

(ii) Each player should play against a new opponent in each round (unless a "bye" is requested in advance).

(iii) The same player cannot have the same colour (black or white) in three successive rounds (i.e., alternate the playing colour of each player as much as possible).

There are many variations to the original algorithm (e.g., FIDE Dubov, FIDE Dutch, FIDE Lim, FIDE Burstein, Amalfi etc. [29]) to address various shortcomings, for example the inability of the algorithm to determine the top $k \geq 2$ players and the lack of clear and widely accepted tie-breaking system for players with the same score at the end of the tournament [19, 36]. Despite this problem, to this day, the Swiss pairing system is used to give out significant monetary prizes in Open tournaments worldwide and determine the official Chess Olympiad results and medals.

Although there is a heuristic rule of thumb proposed in [35] for the required number of rounds to reliably determine the top k players out of n: $R = \frac{(n+7k)}{5}$, in 1972 Haág and Meleghegyi [31] argued in the context of a failed Hungarian National chess tournament that this is not fool-proof due to the negative incentives placed on players and the possibility of draws in chess.

Because the Swiss pairing system has different goals than the algorithm we seek, it is clear that it will not be optimal as is. However, it provides a very useful starting point and raises a number of questions which any pairing algorithm should address, such as how to pair players in the first round, or more generally large groups of players with the same score (*score group*)? How to move a player from a score group with odd number of participants to another (*floaters*)? In what order should score groups be paired?

Before we present and analyze concrete pairing algorithms, we would like to point out some related/modified pairing problems which could be analyzed. In

our current formalism, we allow up to $\lfloor \frac{n}{2} \rfloor$ matches in each restricted round. However, we could impose further restrictions to allow only a maximum of $m \leq \lfloor \frac{n}{2} \rfloor$ matches per round up to the serial case of $m = 1$. A different generalization of the problem is if a single match does not just order 2 players, but up to j of them (e.g., a horse race or swimming competition). Another complexity we may introduce to the model is that of draws in a single match which will result in the possibility of ties in the final ordering of objects. Finally, we may introduce the restrictions (ii) or (iii) from the Swiss system pairing objectives above which would make the formulation more complex, but the resulting algorithm practically more viable.

4 Newly proposed algorithms and results

4.1 Definitions and preliminary observations

At the beginning of the tournament, we assume to have no information about the relative strengths of the players, so any of them could be among the top k, thus they are all *active*. If the rank of a player is definitively determined, the player becomes *inactive*. Below, we restate some results from [42] which we will use in the construction of the proposed algorithms.

Lemma 1 *If a player has played all matches and has* l *losses then his rank is* $l + 1$ *for any* $l \in \{0, 1, \ldots, n - 1\}$.

Lemma 2 *If a player has* w *wins at any point in the tournament then his rank is at most* $n - w$.

Lemma 3 (Carroll [16]) *If a player has* k *losses at any point in the tournament then his rank is at least* $k + 1$ *and will not be among the top* k *players.*

4.2 Combined algorithm and enhancements

Based on the above results, Iványi [42] proposes an algorithm which works on the principle of the Swiss pairing algorithm (i.e., in each round it pairs players with the same or similar scores who have not yet played) and with the following two enhancements:

(i) **Transitivity rule:** At the completion of each round, once the results have been recorded in score matrix M, all of the additional results which can be deduced using the transitivity assumption are recorded.

(ii) **Deletion rule:** If the rank of a player is unambiguously determined then we delete the player and all their results from the score matrix and make them inactive.

Further enhancements can be made to the COMBINED algorithm by clarifying the following set of algorithm attributes:

1. **Ordering of score groups.** When preparing the pairings for the next round, we may begin from the top score group and move the odd player of a score group down to a lower score group (*float down*). Alternatively, we may start from the bottom score group and float up. In formally defining the Swiss pairing algorithm, Olafsson [58] argues for a bi-directional score group order: in the top half of the tournament, the odd player floats down, in the bottom half, they float up and conflicts are iteratively resolved around the middle.

2. **Ordering within a score group.** When determining the "standings" within a score group, for players of the same score, we may decide to simply apply the original indexing of players. (In chess tournaments this is usually based on the ELO rating of the players [23] and represents a pre-conceived strength order). Alternatively, the Buchholz score (sum of the scores of previous opponents) or other tie-breaking score [36] may be computed to determine an ordering within a score group.

3. **Group pairing.** Hollosi and Pahle [35] enlist four different ways in which $2m$ players with a given ordering within a score group may be paired: Fold pairing (1 vs. $2m$, 2 vs. $2m-1, \ldots, m$ vs. $m+1$), Slide pairing (1 vs. $m+1$, 2 vs. $m+2, \ldots, m$ vs. $2m$), Adjacent pairing (1 vs. 2, 3 vs. $4, \ldots, 2m-1$ vs. $2m$) or random. In particular, this method is used to determine the first round pairings for the tournament. For Swiss system chess tournaments, Slide pairing is usually applied.

4. **Handling of unpaired players.** A strict requirement of the Swiss pairing system is that all players must be paired in each round against a new opponent (unless a "bye" has been requested). However, in an optimal partial sorting algorithm this is not a necessary condition. Indeed, when players are deleted, they are omitted from further rounds. There may be pairing situations where active players must remain unpaired, otherwise existing pairings and paired score groups are disrupted. An attribute of any pairing algorithm is how it handles such situations, it may leave

players unpaired or maximize the number of paired players within a score group or combine a score group with another score group in order to maximize the number of paired players [58].

4.3 Top-Down pairing algorithm

Starting from the Combined algorithm of Iványi [42] and considering the algorithm attributes of the previous section, we propose the following Top-Down pairing algorithm. At the beginning of each round, players are sorted as follows:

(i) In increasing order by the number of losses.

(ii) If the number of losses is equal then in decreasing order by their total score.

(iii) If the total score is also equal then in decreasing order by their Buchholz score.

(iv) If all of the above are equal then in increasing order by the initial index.

Once players are sorted, we iterate in a top-down fashion, scanning down the list, trying to find an opponent for the highest ranked unpaired player. If an opponent cannot be found for a player, we leave them unpaired and move to the next unpaired player on the list. At the end of each round, we record the results in the score matrix, fill in the results implied by the transitive rule, update the Buchholz scores and determine if any player can be made inactive and possibly added to the top k players. The formal pseudocode for Top-Down, recorded in the conventions described in Cormen et al. [17] is given below.

Input. n: the total number of players; k: the number of top players to rank; $V = [V_1, V_2, \ldots, V_n]$: the relative strengths of players, a permutation of the numbers 1 to n.

Output. r: the required number of rounds; m: the required number of matches, $res = [res_1, \ldots . res_k]$: ordered list of the index of the top k elements in V.

Work variables. i, j: cycle variables; c: counter for res; M: match matrix, augmented with 7 helping metrics for each player

```
TOP-DOWN(n, k, V)
01 c = 1                          // lines 01–02: initialization of working variables
02 m = 0
03 for i = 1 to n                                  // lines 03–12: initialization of M
04     for j = 1 to n
05         M_{i,j} =null
06     M_{i,n+1} = 0                                          // total score for player i
07     M_{i,n+2} = 0                                        //number of losses for player i
08     M_{i,n+3} = 0                                         // Buchholz score for player i
09     M_{i,n+4} = 0        // flag for having been paired this round for player i
10     M_{i,n+5} = 1                                         // active flag for player i
11     M_{i,n+6} = V[i]                                              // rank for player i
12     M_{i,n+7} = i                                      // original index for player i
13 for r = 1 to n − 1                 // maximum of n − 1 rounds to be paired
14     for i = 1 to n − 1
15         if (M_{i,n+4} == 0 and M_{i,n+5} == 1)    //active and not yet paired
16             for j = i + 1 to n
17                 if (M_{j,n+4} == 0 and M_{j,n+5} == 1 and M_{i,j} == null)
18                         m = m + 1
19                         M_{j,n+4} = 1
20                         M =RecordResult(i, j, M)
21                         break
22     M =UpdateTrans(M)    // lines 22-38 tasks at the end of each round
23     M =UpdateBuchholz(M)
24     for i = 1 to n
25         M_{i,n+4} = 0                                              // reset paired flag
26     M =SortMatrix(M)                 // sort by losses, total score, Buchholz
27     for i = 1 to n − 1                 // lines 27-35 deactivate right players
28         if M_{i,n+5} == 1
29             if M_{i,n+2} < M_{i+1,n+2}
30                 res[c] = M_{i,n+7}
31                 M_{i,n+5} = 0
32                 c = c + 1
33                 if k == c − 1
34                         return r, m, res
35             else break
36     if   c == n                                    // special case for last player
37         res[c] = M_{n,n+7}
38         return r, m, res
```

Note that helper functions RecordResult, UpdateTrans, UpdateBuchhlz and SortMatrix are used in Top-Down, but their pseudocode are not listed here for the sake of brevity. They are constructed in a straightforward way as explained at the beginning of this section.

Top-Down is simple in that it does not consider score groups separately and thus it avoids the problem of floating the odd player in a score group. It is also intuitive, in that it aims to pair the strongest players against the strongest available opponent in a greedy fashion. In order to demonstrate how the algorithm works, we will present a simple example for $n = 4$. Using the notation of [42], let $P_{i,j}$ denote the player with index i and rank j. We will consdier the following permutation of four players: $\pi_1 = \{P_{1,1}, P_{2,4}, P_{3,2}, P_{4,3}\} = \{1, 4, 2, 3\}$ and assume we are interested in finding the full ordering, so $k = 4$.

The basic principle of Top-Down is that once the players are sorted (before the first round, this is done by index), the top player is paired with the highest available opponent, so $P_{1,1}$ is paired against $P_{2,4}$ and the remaining two players are paired against each other. Once the results are recorded, the stylized match matrix including relevant tournament result columns and with the round number in the index of the result is shown in Table 1.

Player	$P_{1,1}$	$P_{2,4}$	$P_{3,2}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_1			1	0	0	1
$P_{2,4}$	0_1	X			0	1	0	1
$P_{3,2}$			X	1_1	1	0	0	1
$P_{4,3}$			0_1	X	0	1	0	1

Table 1: Stylized match matrix M after the first round.

Since there are no transitive results to record, the players are then sorted in increasing order of losses, keeping the index order between tied players, yielding the ordering $P_{1,1}, P_{3,2}, P_{2,4}, P_{4,3}$. Since sorting by total score and Buchholz does not modify this order, the second round top-down pairing will be performed on this ordering. $P_{1,1}$ is now paired against $P_{3,2}$ and $P_{2,4}$ is paired against $P_{4,3}$. Table 2 shows the stylized match matrix with pre-round sorting, after the second round results are recorded, but before the application of the transitivity rule.

Applying the transitivity rule, the results of the matches $P_{1,1}$ vs. $P_{4,3}$ and $P_{3,2}$ vs. $P_{2,4}$ can be deduced and the newly sorted result matrix is shown in Table 3 with results obtained by the transitivity rule shown in bold.

Player	$P_{1,1}$	$P_{3,2}$	$P_{2,4}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_2	1_1		2	0	1	1
$P_{3,2}$	0_2	X		1_1	1	1	1	1
$P_{2,4}$	0_1		X	0_2	0	2	0	1
$P_{4,3}$		0_1	1_2	X	1	1	0	1

Table 2: Stylized match matrix M after the second round.

Player	$P_{1,1}$	$P_{3,2}$	$P_{2,4}$	$P_{4,3}$	Score	Losses	Buchholz	Active
$P_{1,1}$	X	1_2	1_1	$\mathbf{1_2}$	3	0	3	1
$P_{3,2}$	0_2	X	$\mathbf{1_2}$	1_1	2	1	1	1
$P_{4,3}$	$\mathbf{0_2}$	0_1	X	1_2	1	2	0	1
$P_{2,4}$	0_1	$\mathbf{0_2}$	0_2	X	0	3	0	1

Table 3: Stylized match matrix M after two full rounds and the transitivity rule applied.

Applying the de-activation rules of the algorithm, we can see that the full ordering has been determined in 2 rounds. Working through the same algorithm for the other 23 permutations, we observe that in one third of the cases, Top-Down finds the full ranking in 2 rounds, while in two thirds of the cases, 3 rounds are necessary, yielding an average of 2.66667 rounds to be necesesary. Some more results associated with the Top-Down algorithm for small values of n and k are presented in Table 4.

It is trivial that the cases $n = k$ and $n = k-1$ are equivalent, so the former is not even shown in the table. However, further examining the table of results, we observe that there is no difference, on average, between finding the top 2 or 3 amongst 4 players and similarly the top 6 or 7 amongst 8 players. This implies that finding the 2^{nd} best player automatically implies the 3^{rd} best out of 4 and similarly, the same round that determines the 6^{th} best always determines the 7^{th} best out of 8 players. Why these are true, but the same relationship does not hold for $n = 6$ and $k = 4$ and 5 can be the subject of future research.

Conjecture 4 *The* Top-Down *algorithm is optimal amongst deterministic algorithms in terms of the worst case number of rounds required, that is*

$$R(n,k) = \max_i \textsc{Top-Down}(n,k,\Pi_i)\,, \tag{2}$$

where $R(n,k)$ *is as defined in equation (1).*

n	k	min	max	average
2	1	1	1	$2/2 = 1$
4	1	2	2	$48/24 = 2$
4	2	2	3	$64/24 = 2.66667$
4	3	2	3	$64/24 = 2.66667$
6	1	2	3	$2040/720 = 2.83333$
6	2	2	4	$2552/720 = 3.54444$
6	3	2	5	$2808/720 = 3.9$
6	4	2	5	$2960/720 = 4.11111$
6	5	2	5	$2992/720 = 4.15556$
8	1	3	3	$120960/40320 = 3$
8	2	3	5	$160128/40320 = 3.97143$
8	3	3	6	$183296/40320 = 4.54603$
8	4	3	6	$192512/40320 = 4.77460$
8	5	3	6	$200576/40320 = 4.97460$
8	6	3	6	$206464/40320 = 5.12063$
8	7	3	6	$206464/40320 = 5.12063$

Table 4: Minimal, maximal and average number of rounds required for Top-Down for small values of n and k.

Note that we do not conjecture optimality for the number of matches, as Top-Down pairs all the players it can in each round even though in the last round this may not be necessary for the determination of the k best players.

4.4 Exhaustive pairing algorithm

In this subsection, we will assume that n is even and consider all possible, unique pairings of n elements. In the case that n is odd, we can make it even by adding a dummy element which is smaller than all other elements. Now, consider all possible pairings for the first round. After any pairing in the first round, there will be exactly $\frac{n}{2}$ winners and $\frac{n}{2}$ losers. We could then consider all possible pairings for the second round and evaluate the standings after applying the transitive and deletion rules. We could continue this in a brute-force exhaustive way and for a given input permutation of elements, find the best possible pairing which determines the top k elements most quickly.

For the trivial case of $n = 2$, a single round with the one possible pairing completes the full ranking. The following result gives a constructive solution to the exhaustive approach for all even $n \geq 4$.

Theorem 5 *For any even $n \geq 4$ there exists a pairing which, utilizing the transitivity rule, determines the full ordering of n players in exactly 2 rounds.*

Proof. Let P_i be the index of the player with rank i for all $i \in \{1, \ldots, n\}$. We first observe that the match between P_i and P_{i+1} must be played for all i, $1 \leq i \leq k-1$ because these results cannot be deduced by transitivity. Therefore, at least $k-1$ matches will be necessary to determine the top k players, or $n-1$ matches in the case $k = n$. Since at most $\frac{n}{2}$ matches can be played in one round, at least 2 rounds are needed to construct full ordering.

Now, consider the following pairing which shows that this lower bound is tight:
Round 1: $\{P_1 - P_2, P_3 - P_4, \ldots, P_{n-1} - P_n\}$. The player listed first wins each match.
Round 2: $\{P_2 - P_3, P_4 - P_5, \ldots, P_{n-2} - P_{n-1}\}$. The player listed first again wins each match.
Applying the transitivity rule, we can reconstruct the full ordering and determine the top k players $\{P_1, \ldots, P_k\}$ for any k. □

The above result demonstrates that pairing players with different scores can be efficient, if the player with the lower score wins. This observation could inspire randomly introducing such pairings into the algorithm, which would lead to stochastic pairing algorithms, but this is beyond the scope of this paper.

5 Conclusions and directions for future research

In this paper, we introduced the problem of partial sorting in restricted rounds, where in each round, each element can only be compared with at most one other element. We examined various algorithms for minimizing the number of rounds required to select the top k elements and made a conjecture about Top-Down being optimal among the class of deterministic algorithms. Further computer simulations and comparison against various benchmarks and ultimately proving this conjecture mathematically is a fertile area of future research.

We also considered exhaustively finding the best possible pairing among all possible pairings and proved that for any n and any permutation, there exists

a pairing which determines the top k elements in just 2 rounds for any k. This approach shows that pairing players with the same score is not necessarily optimal and points towards considering stochastic pairing algorithms. Sadly, professor Iványi could not complete this, his last project, so this paper is dedicated to his memory.

Acknowledgements

The authors would like to thank Gyula Császár, Matej Jusup, László Csató and Ervin Haág for useful consultations and discussions.

References

[1] M. Aigner, Selecting the top three elements, *Discrete Appl. Math.*, **4** (1982) 242–262. ⇒18

[2] M. Aigner, The double selection problem, *Discrete Math.*, **73** (1989) 3–12. ⇒18

[3] M. Aigner, Finding the maximum and minimum, *Discrete Appl. Math.*, **97** (1997) 1–12. ⇒18

[4] M. Ajtai, J. Komlos, E. Szemeredi, Sorting in $\mathcal{O}(\log \mathtt{n})$ steps, *Combinatorica*, **3,** 1 (1983) 1–19. ⇒19

[5] N. Alon, Y. Azar, U. Vishkin, Tight complexity bounds for parallel comparison sorting, *IEEE Symp. Found. Comp. Sci.*, **27** (1986) 502–510. ⇒19

[6] M. Anholcer, V. Babiy, S. Bozóki, W. W. Koczkodaj, A simplified implementation of the least squares solution for pairwise comparisons matrices. *CEJOR Cent. Eur. J. Oper. Res.* **19,** 4 (2011) 439–444. ⇒18

[7] M. Ayala-Rincón, B. T. de Abreu, J. de Sequira, A variant of the Ford-Johnson algorithm that is more space efficient. *Inf. Proc. Letters*, **102,** 5 (2007) 201–207. ⇒18

[8] H. Aziz, M. Brill, F. Fischer, P. Harrenstein, J. Lang, H. G. Seedig, Possible and necessary winners of partial tournaments, in: V. Conitzer and M. Winikoff (eds.), *Proc. of 11th Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (AAMAS), IFAAMAS, 2012. 8 pages. ⇒19

[9] L. B. Beasley, D. E. Brown, K. B. Reid, Extending partial tournaments, *Math. Comput. Modelling* **50,** 1 (2009) 287–291. ⇒19

[10] M. Blum, R. W. Floyd, W. Pratt, R. L. Rivest, R. E. Tarjan, Time bounds for selection, *J. Computer System Sci.*, **7** (1973) 464–471. ⇒18

[11] B. Bollobás, A. Thomason, Parallel sorting, *Discrete Appl. Math.*, **6,** 1 (1983) 1–11 ⇒19

[12] B. Bollobás, P. Hell, Sorting and graphs, in: *Graphs and Order* (ed. I. Rival), Reidel, Boston, 1985, pp.169–184. ⇒19

[13] S. Bozóki, J. Fülöp, A. Poesz, On pairwise comparison matrices that can be made consistent by the modification of a few elements, *CEJOR Cent. Eur. J. Oper. Res.* **19** (2011) 157–175. ⇒18

[14] S. Bozóki, J. Fülöp, L. Rónyai, On optimal completion of incomplete pairwise comparison matrices, *Math. Comput. Modelling* **52** (2010) 318–333. ⇒18

[15] T. D. Bui, M. Thanh, Significant improvements to the Ford-Johnson algorithm for sorting, *BIT*, **25** (1985) 70–75. ⇒18

[16] L. Carroll, Lawn tennis tournaments, *St. James' Gazette*, August 1, 1883, 5–6. Reprinted in *The Complete Works of Lewis Carroll*, Newyork Modern Library, 1947. ⇒20, 22

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms* (3rd edition), The MIT Press, 2009. ⇒24

[18] L. Csató, Ranking by pairwise comparisons for Swiss-system tournaments, *Cent. Eur. J. Oper. Res.*, **21,** 4 (2013) 783–803. ⇒18

[19] L. Csató, On the ranking of a Swiss system, chess team tournament, *Annals of Op. Res.,* **254,** 1–2 (2017) 17–36. ⇒18, 21

[20] W. Cunto, J. I. Munro, Average case selection, *J. ACM*, **36,** 2 (1989) 270–279. ⇒18

[21] D. Dor, J. Hástad, S. Ulfberg, U. Zwick, On lower bounds for selecting the median, *SIAM J. Discrete Math.*, **14,** 3 (2001) 299–311. ⇒18

[22] D. Dor, U. Zwick, Selecting the median, *SIAM J. Comp.*, **7**, 5 (1999) 1722–1758. ⇒18

[23] A. E. Elo, *The Rating of Chessplayers, Past and Present*, Batsford, London, 1978. ⇒23

[24] J. Eusterbrock, Errata to "Selecting the top three elements" by M. Aigner, *Discrete Appl. Math.*, **41** (1993) 131–137. ⇒18

[25] FIDE, *Handbook. 04. FIDE Swiss rules,* 2013, http://www.fide.com/component/handbook/?id=83&view=article, downloaded June 6, 2017. ⇒21

[26] FIDE, *Basic rules for Swiss Systems*, http://www.fide.com/fide/handbook.html?id=83&view=article, downloaded June 6, 2017. ⇒21

[27] FIDE, *Dutch System* https://www.fide.com/fide/handbook.html?id=167&view=article, downloaded June 6, 2017. ⇒21

[28] L. Ford, S. Johnson, A tournament problem, *Amer. Math. Monthly* **66** (1959) 387–389. ⇒18

[29] L. Forlano, *VEGA chess pairing software, User's manual* http://www.vegachess.com/tl/tl_files/music_academy/distrib/vega_en.pdf, downloaded June 6, 2017. ⇒21

[30] J. Griggs, K. B. Reid, Landau's theorem revisited, *Australas. J. Comb.* **20** (1999), 19–24. ⇒18

[31] E. Haág, Cs. Meleghegyi, A semifinal that decided nothing (Hungarian), *Magyar Sakkélet* 1972 (10), 190–191. ⇒21

[32] A. Hadian, M. Sobel, Selecting the t^{th} largest using binary errorless comparisons. TR No. 121, Univ. of Minnesota, Department of Statistics, 1969. ⇒18
[33] S. L. Hakimi, On the realizability of a set of integers as degrees of the vertices of a simple graph. *J. SIAM Appl. Math.* **10** (1962) 496–506. ⇒18
[34] R. Haggkvist, P. Hell, Parallel sorting with constant time for comparisons, *SIAM J Comput*, **10,** 3 (1981) 465–472. ⇒19
[35] A. Hollosi, M. Pahle, Swiss pairing, in *Sensei's Library,* Graz, 2013, http://senseis.xmp.net/?SwissPairing, Downloaded June 6, 2017. ⇒21, 23
[36] A. Hollosi, M. Pahle, Tie Breaker, in *Sensei's Library,* Graz, 2013, http://senseis.xmp.net/?SwissPairing, Downloaded June 6, 2017. ⇒21, 23
[37] A. Iványi, Reconstruction of complete interval tournaments, *Acta Univ. Sapientiae, Inform.*, **1,** 1 (2009) 71–88. ⇒18
[38] A. Iványi, Reconstruction of complete interval tournaments II., *Acta Univ. Sapientiae, Math.*, **2,** 1 (2010) 47–71. ⇒18
[39] A. Iványi, Directed graphs with prescribed score sequences, in: *The 7th Hungarian-Japanese Symposium on Discrete Mathematics and Applications* (ed. S. Iwata, Kyoto, May 31 - June 3, 2011), 114–123. ⇒18
[40] A. Iványi, Deciding football sequences, *Acta Univ. Sapientiae, Inform.*, **4,** 1 (2012) 130—183. ⇒18
[41] A. Iványi, Degree sequences of multigraphs. *Annales Univ. Sci. Budapest., Sect. Comp.* **37** (2012) 195–214. ⇒18
[42] A. Iványi, Z. Kása, Parallel partial ranking, *Appl. Discr. Math. and Heur. Alg.*, **1,** 3 (2015) 57–76. ⇒22, 24, 26
[43] A. Iványi, L. Lucz, T. F. Móri, P. Sótér, On the Erdős-Gallai and Havel-Hakimi algorithms. *Acta Univ. Sapientiae, Inform.* **3,** 2 (2011) 230–268. ⇒18
[44] A. Iványi, S. Pirzada, Comparison based ranking, in: *Algorithms of Informatics, Vol. 3* (ed. A. Iványi), AnTonCom, Budapest 2011, 1209–1258. ⇒18
[45] G. Kéri, On qualitatively consistent, transitive and contradictory judgment matrices emerging from multiattribute decision procedures, *CEJOR Cent. Eur. J. Oper. Res.* **19,** 2 (2011) 215–224. ⇒18
[46] H. Kim, Z. Toroczkai, I. Miklós, P. L. Erdős, L. A. Székely, Degree-based graph construction, *J. Physics*: *Math. Theor. A* **42,** 39 (2009), 392001-1-3920001.10. ⇒18
[47] D. G. Kirkpatrick, A unified lower bound for selection and set partitioning problems, *J. ACM*, **28** (1981) 150–165. ⇒18
[48] D. G. Kirkpatrick, Closing a long-standing complexity gap for selection: $V_3(42) = 50$, in *Space-efficient Data Structures, Streams, and Algorithms*, Springer Verlag, Berlin, 2013, pp.61–76 ⇒18
[49] S.S. Kislitsyn, Finding the k^{th} element in ordered set with pairwise comparisons (in Russian), Sibirsk. Mat. Zh., **2,** 5 (1964) 557–564. ⇒18
[50] D. E. Knuth, *The Art of Computer programming, Vol. 3. Sorting*, Addison-Wesley, Upper Saddle River, NJ, 1998. ⇒18, 20

[51] H. G. Landau, On dominance relations and the structure of animal societies. III. The condition for a score sequence, *Bull. Math. Biophys.* **15** (1953) 143–148. ⇒ 18

[52] F. Liljeros, C. R. Edling, L. Amaral, H. E. Stanley, Y. Aberg, The web of human sexual contacts. *Nature* **411** (2001) 907–908. ⇒18

[53] G. K. Manacher, The Ford-Johnson sorting algorithm is not optimal. *J. ACM*, **26,** 3 (1979) 441–456. ⇒18

[54] G. K. Manacher, Significant improvements to the Hwang-Lin merging algorithm, *J. ACM*, **26,** 3 (1979) 434–440. ⇒18

[55] G. K. Manacher, T. D. Bu, T. Mai, Optimal combinations of sorting and merging, *J. ACM* **36** (1989) 290–334. ⇒18

[56] C. Martínez, Partial quicksort *Proc. 6th ACM-SIAM Workshop on Algorithm Engineering and Experiments and 1st ACM-SIAM Workshop on Analytic Algorithmics and Combinatorics.* (2004) 5 pages. ⇒18

[57] M. Newman, A. L. Barabási, D. J. Watts, *The Structure and Dynamics of Networks.* Princeton University Press, (2006). ⇒18

[58] S. Ólaffson, Weighted matchings in chess tournaments, *J. Oper. Res. Soc.*, **41,** 1 (1990) 17–24. ⇒21, 23, 24

[59] M. Peczarski, The Ford-Johnson algorithm still unbeaten for less than 47 elements, *Inf. Proc. Letters*, **101,** 3 (2007) 126–128. ⇒18

[60] M. Peczarski, Towards optimal sorting of 16 elements. *Acta Univ. Sapientiae, Inform.* **4,** 2 (2012) 215–224. ⇒18

[61] N. Pippenger, Sorting and selecting in rounds, *SIAM J Comput*, **16,** 6 (1987) 1032–1038. ⇒18

[62] S. Pirzada, A. Iványi, Minimal digraphs with given imbalance sets, *Acta Univ. Sapientiae, Math.*, **4,** 1 (2012) 86–101. ⇒18

[63] I. Pohl, A sorting problem and its complexity, *Comm. ACM* **15,** 6 (1972) 462–464. ⇒18

[64] K. B. Reid, Tournaments: Scores, kings, generalizations and special topics, *Congr. Numer.* **115** (1996) 171–211. ⇒18

[65] K. B. Reid, C. Q. Zhang, Score sequences of semicomplete digraphs, *Bull. Inst. Combin. Appl.* **24** (1998) 27–32. ⇒18

[66] A. Schönhage, M. Paterson, N. Pippenger, Finding the median, *J. Comp. Syst. Sci.*, **13** 2 (1976) 184–199. ⇒18

[67] J. Schreier, The tournament elimination systems, *Mathesis Polska*, (7) (1932) 154–160. ⇒18

[68] L. Slupecki, On the systems of tournaments, *Colloquium Math.* **2,** 4 (1951) 286–290. ⇒18

[69] J. Temesi, L. Csató, S. Bozóki, Tennis of old times and today. An application of the partially filled comparison matrices (Hungarian), in: (ed. T. Solymosi and F. Forgó) *Balance and optimum. Case studies for the seventieth birthday of Ferenc Forgó*, Aula, Budapest, 2012, 213–245. ⇒18

[70] Á. Varecza, On the largest and smallest elements. *Ann. Univ. Sci. Budapest. Sect. Comput.*, **4** (1983) 3–10. ⇒18

[71] A. C. Yao, F. F. Yao, On the average-case complexity of selecting the k^{th} best, *SIAM J. Comp.*, **11,** 3 (1982) 428-447. ⇒18
[72] C. K. Yap, New upper bounds for selection, *Comm. ACM*, **19,** 9 (1976) 501–508. ⇒18

Received: June 12, 2017 • Revised: July 3, 2017