POLITECNICO DI TORINO

1859 1906

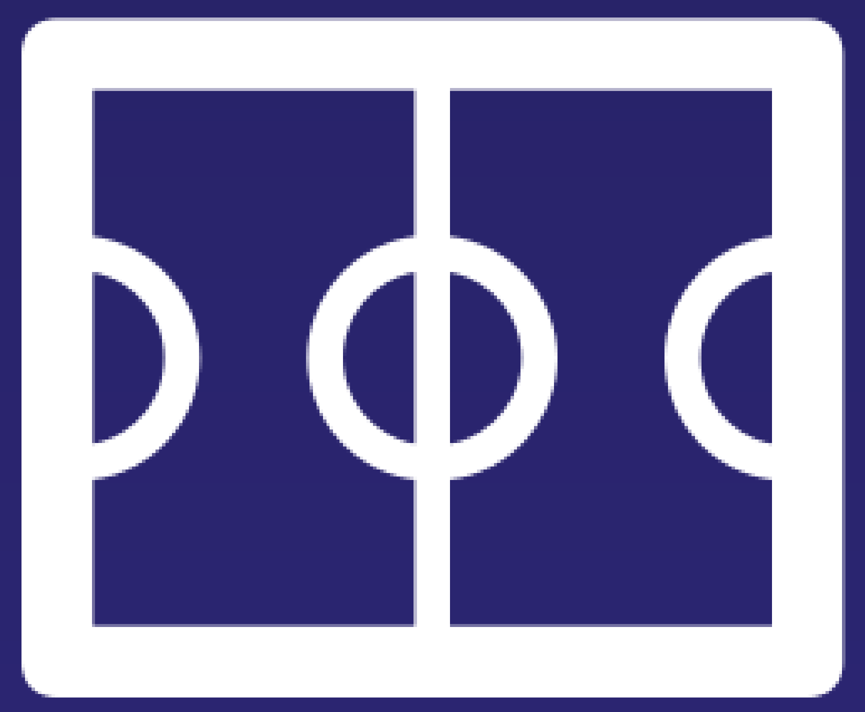
ALCO

Fairness players allocation in ATP Tournaments generation

Table of contents

Background

Background



Your tennis career

- Active dedication
- Hard training & Diet
- Hundreds of Tournaments
- Economical effort

ATP GRAND SLAMS
ATP 250-500-1000
ATP CHALLENGER
ITF MEN

RANKING [1,75]

AUG 29, 2017 @ 08:30 AM 48,471 EDITOR'S PICK The Little Black Book of Billionaire Secrets

Roger Federer Heads The World's Highest-Paid Tennis Players Of 2017

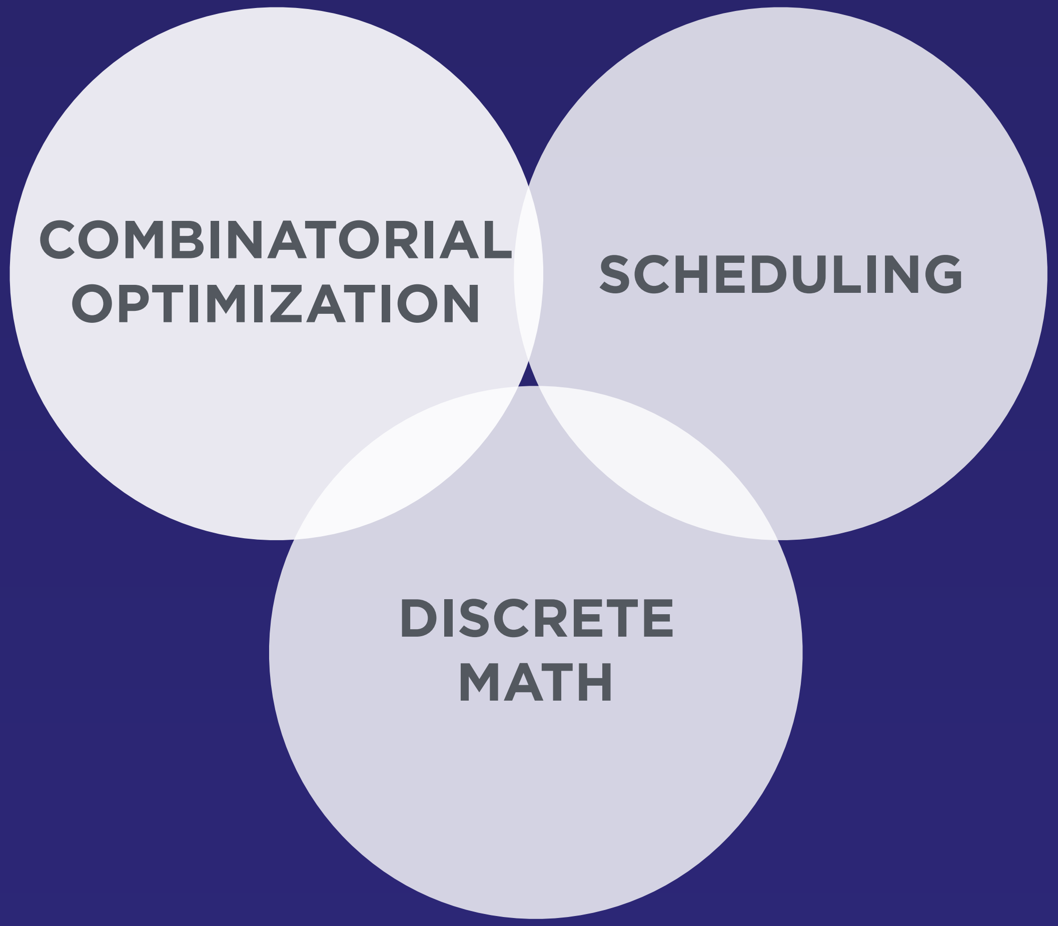
The rest

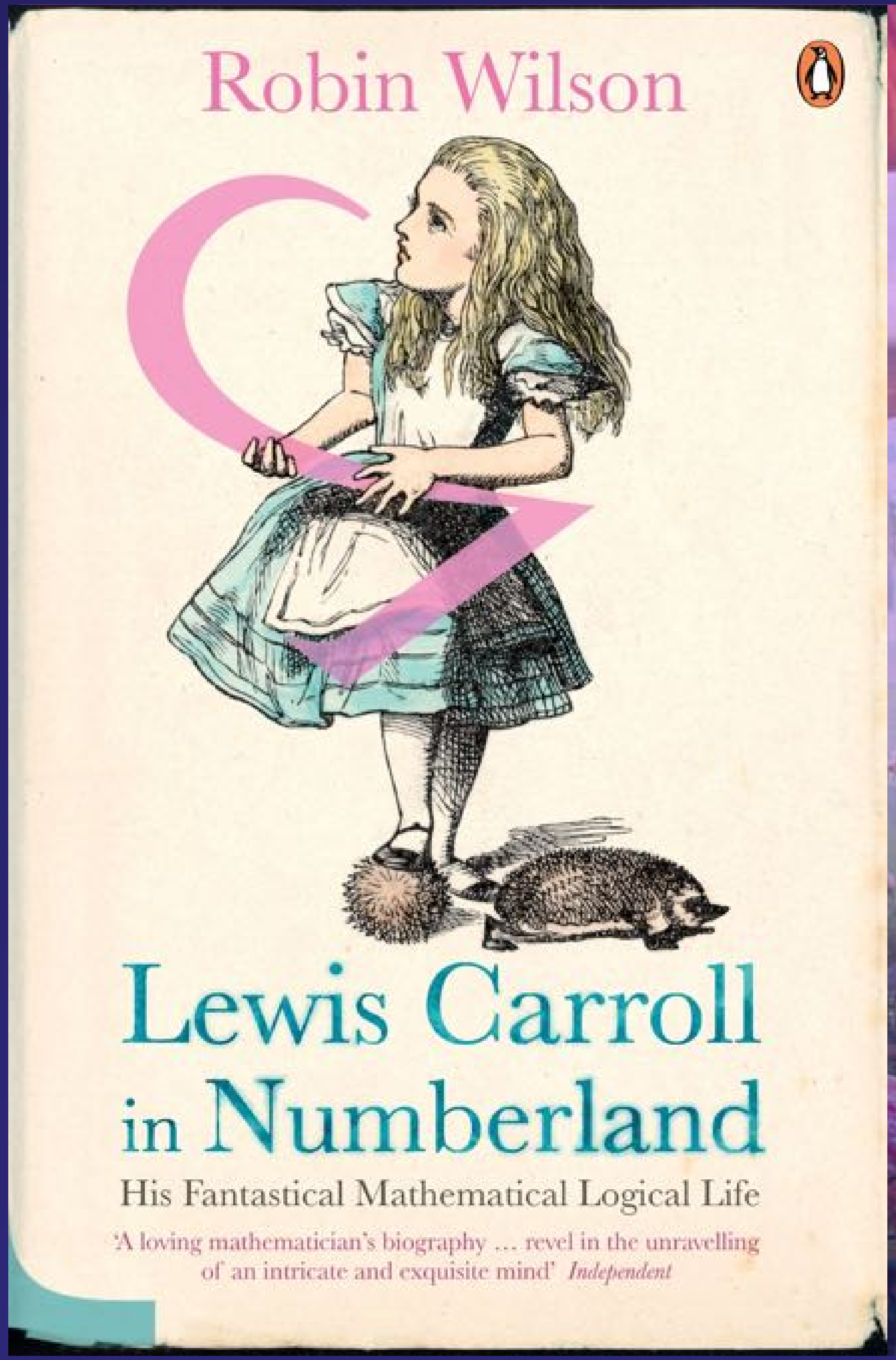
AUG 26, 2013 @ 08:28 AM 196,913 The Little Black Book of Billionaire Secrets

How The 92nd-Ranked Tennis Player In The World Earns A Comfortable Living

Single elimination

The single elimination is a type of tournament in which **the loser** of each match is **directly eliminated** from the game, while the **winner moves on** to the next round.



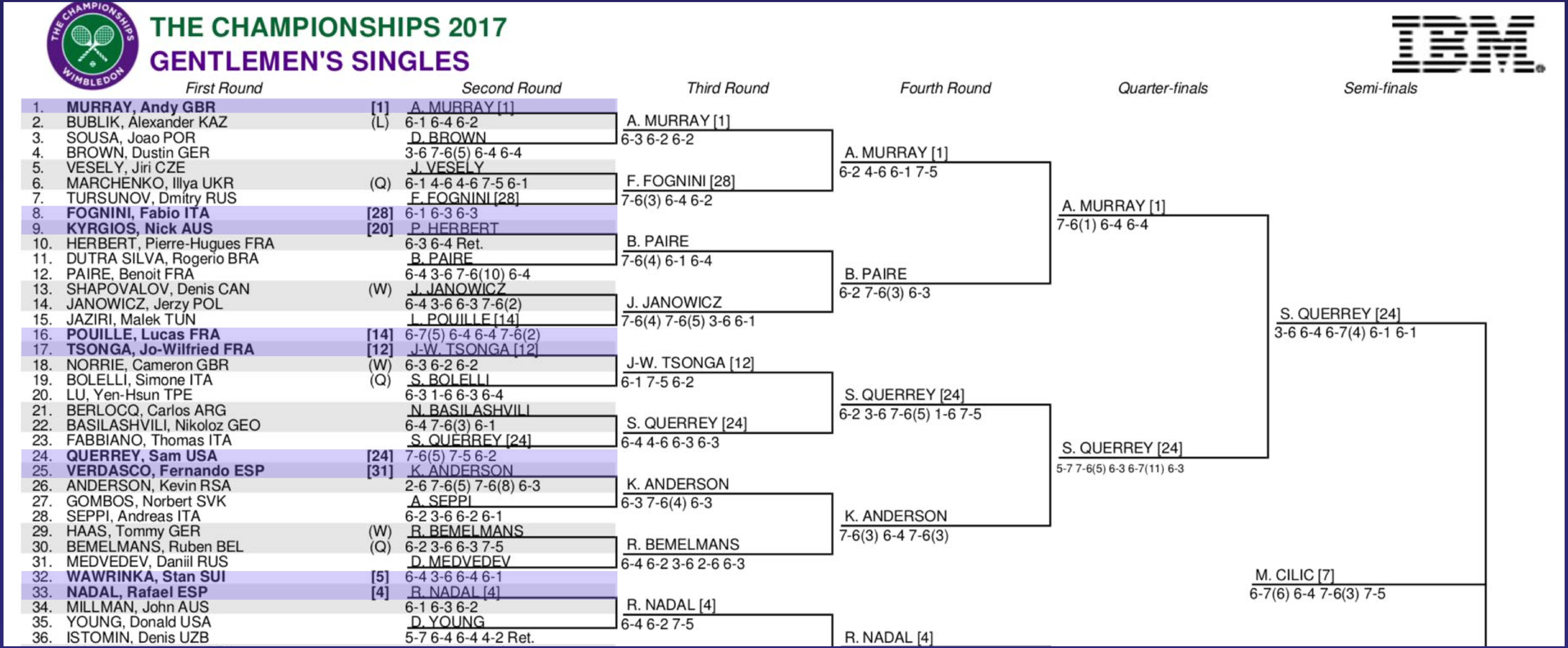


Players are allocated according to the ***luck of the draw***. This may lead to a scenario in which top players are competing against each other in early rounds.

Seeding is a technique for allocating - in a given tournament - an amount of top players so that they will possibly play against other top players only later on in the tournament.

Brackets graph

Background



Fairness in Tennis

An overview

Different **configurations** for a generic tournament lead to diverse patterns of **winners and losers.**
(Horen and Riezman, 1985)

Under certain assumptions - there is always a **specific tournament structure** which **maximizes the odds of winning** for any generic player
(Williams, 2010)

World Cup draw: quantifying (un)fairness and (im)balance: the **FIFA world cup** 2010.
(Guyon, 2010)

Operations Research Transformed the **Scheduling of South American World Cup** Qualifiers
(Duràn et al, 2017)

What can be then fairness in Tennis?

Match repetitions

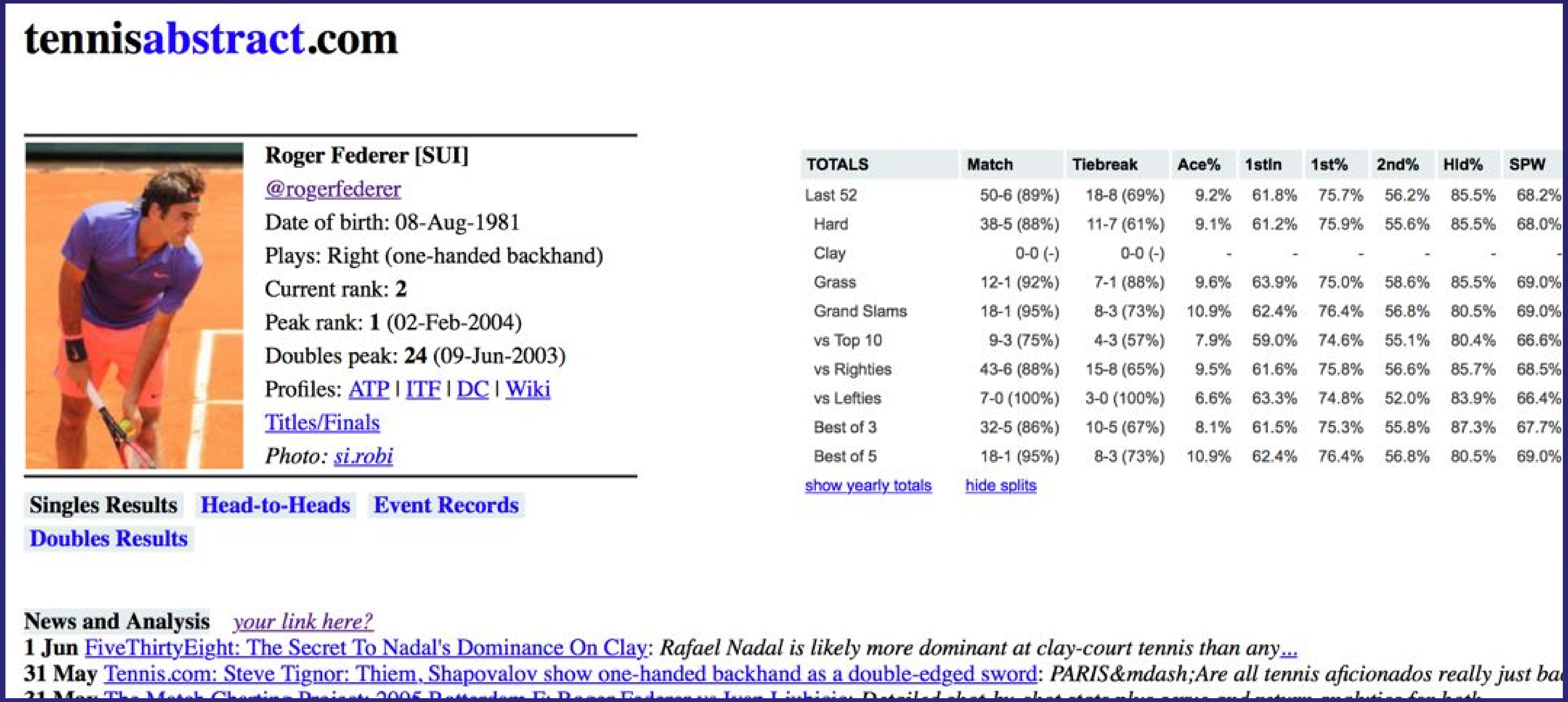
There are several cases in which players have been **paired (1ST ROUND) with the same opponent multiple times** during a **time-window** ranging from 1 to 3 months.

DECREASING INTEREST

VARIETY OF MATCHES

Trivia: Deja Vu All Over Again

Italian translation at settesei.it



tennisabstract.com

Roger Federer [SUI]
@rogerfederer
Date of birth: 08-Aug-1981
Plays: Right (one-handed backhand)
Current rank: **2**
Peak rank: **1** (02-Feb-2004)
Doubles peak: **24** (09-Jun-2003)
Profiles: ATP | ITF | DC | Wiki
Titles/Finals
Photo: si.robi

TOTALS	Match	Tiebreak	Ace%	1stIn	1st%	2nd%	Hld%	SPW
Last 52	50-6 (89%)	18-8 (69%)	9.2%	61.8%	75.7%	56.2%	85.5%	68.2%
Hard	38-5 (88%)	11-7 (61%)	9.1%	61.2%	75.9%	55.6%	85.5%	68.0%
Clay	0-0 (-)	0-0 (-)	-	-	-	-	-	-
Grass	12-1 (92%)	7-1 (88%)	9.6%	63.9%	75.0%	58.6%	85.5%	69.0%
Grand Slams	18-1 (95%)	8-3 (73%)	10.9%	62.4%	76.4%	56.8%	80.5%	69.0%
vs Top 10	9-3 (75%)	4-3 (57%)	7.9%	59.0%	74.6%	55.1%	80.4%	66.6%
vs Righties	43-6 (88%)	15-8 (65%)	9.5%	61.6%	75.8%	56.6%	85.7%	68.5%
vs Lefties	7-0 (100%)	3-0 (100%)	6.6%	63.3%	74.8%	52.0%	83.9%	66.4%
Best of 3	32-5 (86%)	10-5 (67%)	8.1%	61.5%	75.3%	55.8%	87.3%	67.7%
Best of 5	18-1 (95%)	8-3 (73%)	10.9%	62.4%	76.4%	56.8%	80.5%	69.0%

show yearly totals   hide splits

Singles Results **Head-to-Heads** **Event Records**
Doubles Results

News and Analysis *your link here?*
1 Jun FiveThirtyEight: The Secret To Nadal's Dominance On Clay: *Rafael Nadal is likely more dominant at clay-court tennis than any...*
31 May Tennis.com: Steve Tignor: Thiem, Shapovalov show one-handed backhand as a double-edged sword: *PARIS—Are all tennis aficionados really just ba*

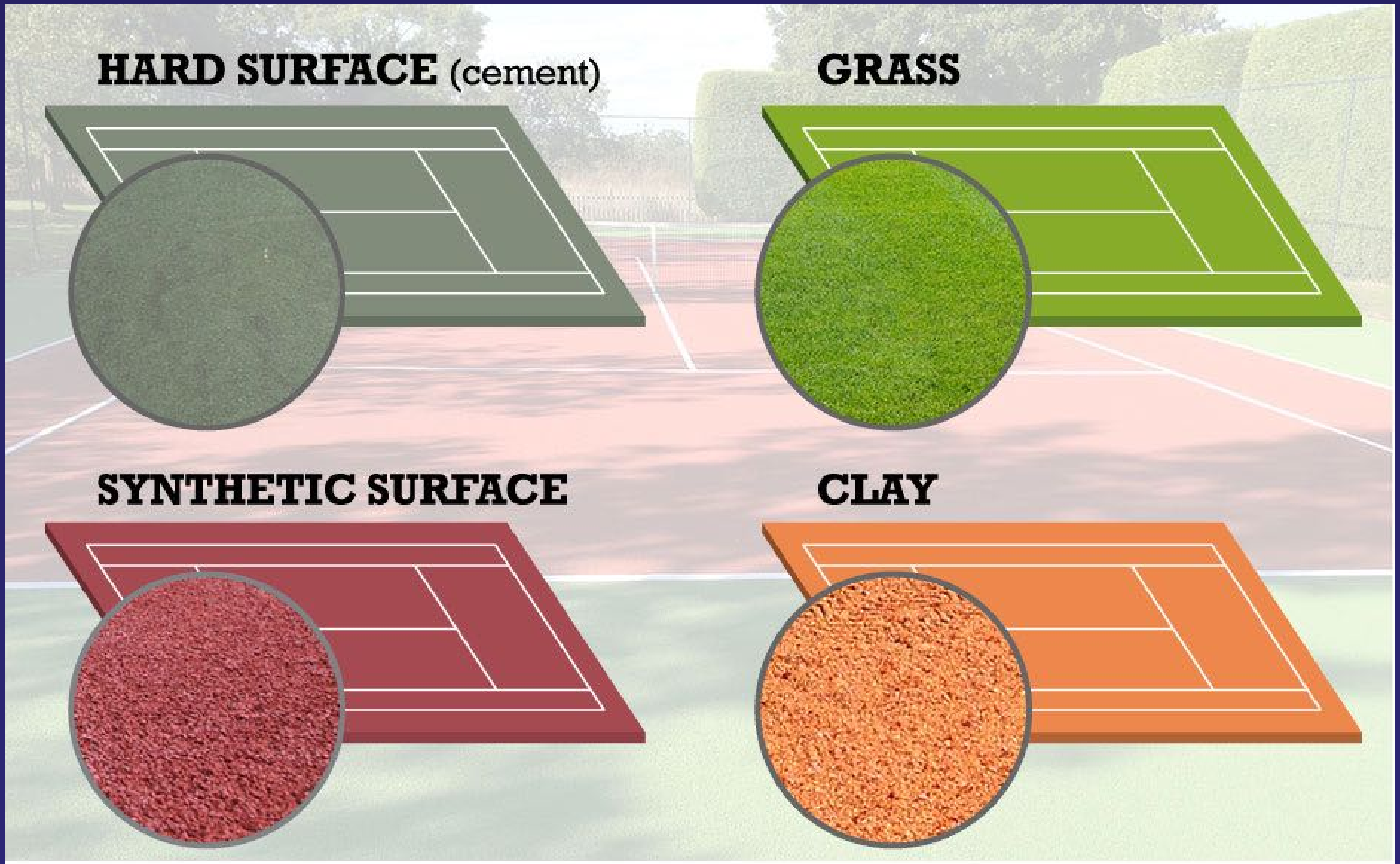
Other conflicts

Some side constraints might be required in certain tournaments. For instance, some organisers may require to pair only players of **different nationalities.**

SURFACE

NATIONALITY

EVERYTHING YOU CAN THINK OF



(

Repetitions against seeded

An unseeded player (*rank > 75*) paired with a seeded player for two tournaments in a row might experience an **noticeable damage** both financially and in his career.

CAREER DAMAGE

SEEDED vs UNSEEDED

DECREASING INTEREST

Unseeded Serena and the Roland Garros Draw

In a wide-open women's field at this year's French Open, it seems fitting

Numbers: unlucky players

We define ***unlucky*** a generic *unseeded* player who is paired against two *seeded* players in *2 consecutive tournaments*.

Let's compute the probability that there is **at least one unlucky player** in **2 consecutive tournaments.**

Numbers: unlucky players

n	128	Number of players in the tournament
m	32	Number of **seeded** players
a=n-m	96	Number of **unseeded** players
b=m	32	Number of unseeded players paired with seeds in the first tournament

$$P_{atleast} = 1 - P_{none} = 1 - \frac{\binom{a-b}{b}}{\binom{a}{b}} = 1 - \frac{(a-b)!^2}{a!(a-2b)!}$$

n=128 m=32 P=0.99

n=64 m=16 P=0.99

n=64 m=8 P=0.73

Numbers: unlucky players

ROL-WIM	**7** Unlucky players
WIM-AUS	**6** Unlucky players
AUS-US	**3** Unlucky players

NAME	**RANK**	**WIM**	**ROLGAR**
Andrey Kuznetsov	**73**	Andy Murray (1)	Karen Khachanov (30)
Jordan Thompson	**92**	John Isner (21)	Albert Ramos (25)
Jan Lennard Struff	47	Tomas Berdych (13)	Milos Raonic (6)
Thanasi Kokkinakis	**Q**	Kei Nishikori (8)	Juan Martin Del Potro (29)
Philipp Kohlschreiber	43	Nick Kyrgios (18)	Marin Cilic (7)
John Millman	**142**	Roberto Bautista Agut (17)	Rafael Nadal (4)
Bernard Tomic	39	Dominic Thiem (6)	Mischa Zverev (27)

Andrey Kuznetsov

2017 Grand Slams

Roland Garros	Wimbledon	US Open	AUS Open
Andy Murray	Karen Khachanov	Feliciano Lopez	Kei Nishikori
1	30	31	5

"If I hadn't been a Tennis players, I would have been a singer. But I cannot sing."

Andrey Kuznetsov

Misfortune of unseeded players

Occurrence of matches with seeds in the four Slams of 2017

Occurr.	# Players	%
0/4	**32**	**33,33 %**
1/4	42	43,75 %
2/4	15	15,63 %
3/4	6	6,25 %
4/4	**1**	**1,04 %**

There is a but...

The *luck of the draw*

Single- elimination tournaments are structured by **randomly** assigning a number of players in their respective slots as well as performing a **constrained draw for seeded players**.

Therefore, is essential to ensure that any structuring process involves a randomized draw.

Definitions and formulation

The goal

Create a tournament structure - a *bracket graph* - that:

MINIMIZE MATCH REPETITIONS

MINIMIZE UNLUCKY PLAYERS

REDUCE CONFLICTS

Given a set of **conflicts** - things we *would like to avoid* - we try to split the players into **k different groups** so that the ***measure*** of the conflicts inside each groups is **minimized**.

=

Cluster elements of the set **of players** into **k** buckets.
Then, we perform a constrained **draw in each cluster.**

The idea

Definitions and formulation

A

CONTAINS 25% OF UNSEEDED

CONTAINS 25% OF SEEDED

SLOTS FROM 0 TO 25% ROUND1

B

.
.
.

Assuming the **seeds** are already **allocated**

C

D

CONTAINS 25% OF UNSEEDED

CONTAINS 25% OF SEEDED

SLOTS FROM 75% TO 100% ROUND1

Definitions

$n = 2^t$ Number of players in the tournament

$I = \{i \in \mathbb{N} : 1 \leq i \leq n\}$

$m = 2^p$ Number of seeded players in the tournament.

$M \in I \qquad p < t$

$t \in \mathbb{N}$ Number of rounds. A round is a set of matches between **z** players which results in **z/2** winners

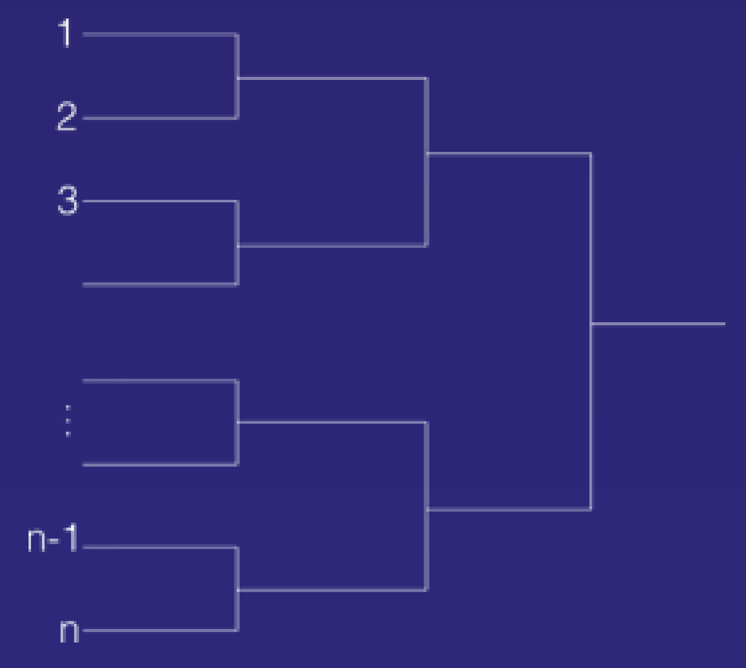
$R = \{r_i \in \mathbb{Z} : 0 \leq r_i < t\}$

$s \in \mathbb{N}$ The structure of a tournament is a mapping that assigns each player ***i*** ∈ ***I*** to a single slot in **S**.

$S = \{s_i \in \mathbb{N} : 1 \leq s_i \leq 2(n-1)\}$

A generic round **r_i** has slot numbers which can be indexed as

$$\mathrm{s}_i \in [1 + \sum_{l=0 \wedge r_i > 0}^{r_i - 1} 2^{rounds-l}, \sum_{l=0}^{r_i} 2^{rounds-l}]$$



Definitions

$k \in \mathbb{N}$ Number of clusters or *buckets* $J = \{j \in \mathbb{N} : 1 \leq j \leq k\}$

$(n \ mod \ k) = 0$

$H^{n \times n}$ Matrix H define the *measure* for all the mutual conflict between players. The generic element $h_{\alpha\beta}$ represents the measure of conflict between α and β. $H^{n \times n} \wedge h_{\alpha,\beta} \in [0, \infty)$

All the conflicts considered are **potential.** A conflict becomes **active** when two player with $h_{\alpha\beta}>0$ are paired together.

The *tournament allocation problem* clusters **n players** with **m seeds** in **k groups**, in order to **minimize match repetitions** and **potential conflicts**. Matches within clusters are then **randomly determined by a draw.**

$$x_{ij} = 1$$

Player ***i*** is allocated to cluster **j**

0/1 Mathematical model

Definitions and formulation

Each player can be allocated to **one and one** cluster.

$$\sum_{j=1}^{k} x_{ij} = 1 \ \ \forall i \in I$$

Each cluster has exactly **u** players.

$$\sum_{i=1}^{n} x_{ij} = u \ \ \forall j \in J$$

$$u = \frac{n}{k}$$

Variables are binary (0/1)

$$x_{ij} \in \{0,1\} \ \ \forall i \in I, j \in J$$
$$\alpha, \beta, i \in I, \ \ j \in J$$

$$minimize\ Z = \sum_{j=1}^{k} (\sum_{\alpha=1}^{n-1} \sum_{\beta=\alpha+1}^{n} h_{\alpha\beta} x_{\alpha j} x_{\beta j})$$

0/1 Mathematical model

Definitions and formulation

$$minimize\ Z = \sum_{j=1}^{k} (\sum_{\alpha=1}^{n-1} \sum_{\beta=\alpha+1}^{n} h_{\alpha\beta} x_{\alpha j} x_{\beta j})$$

DoF

With a first approximation, we can fix **m** variables - corresponding to the *seeded players* - in their clusters. In fact, we assume the seeded players are given. Therefore $DoF = k \cdot (n - m)$

H Matrix

The behaviour heavily depends on the allocation of ***h*** coefficients. We assumed some **arbitrary values** for our simulations.

QUADRATIC

Since two integer variables are multiplied, the model is quadratic. Even thought it can be linearized, **Ilog CPLEX 12.7** does it by default.

0/1 LP model

Definitions and formulation

$$y_{ij} = x_i x_j \wedge y_{ij} \in \{0,1\}$$

So that

$$x_i \geq y_{ij}$$
$$x_j \geq y_{ij}$$
$$x_i + x_j \leq 1 + y_{ij}$$

(Della Croce et al., 2014)

$$minimize\ Z = \sum_{j=1}^{k} \left(\sum_{\alpha=1}^{n-1} \sum_{\beta=\alpha+1}^{n} h_{\alpha\beta} y_{\alpha j,\beta j} \right)$$

$$y_{\alpha j,\beta j} = x_{\alpha j} x_{\beta j}$$

$$\sum_{j=1}^{k} x_{ij} = 1 \;\; \forall i \in I$$

$$\sum_{i=1}^{n} x_{ij} = u \;\; \forall j \in J$$

$$x_{\alpha j} \geq y_{\alpha j,\beta j}$$
$$x_{\beta j} \geq y_{\alpha j,\beta j}$$
$$x_{\alpha j} + x_{\beta j} \leq y_{\alpha j,\beta j} + 1$$
$$y_{\alpha j,\beta j} \in \{0,1\} \; \forall i \in I, j \in J$$

$$x_{ij} = 1 \; \forall i \in M$$
$$x_{ij} \in \{0,1\} \;\; \forall i \in I, j \in J$$
$$\alpha, \beta, i \in I, \;\; j \in J$$

Related CO problems

Max cut problem

Given a generic graph ***G = (V,E)*** where V is the set of nodes and E the one of edges, a **cut** is defined as a **partition of V** into **two disjoint subset.**
The *weight of the cut* is defined as the sum of edges' weights which have **endpoints** in the two different subsets.

The max cut problem seeks to **cluster** nodes in **V** in two different subsets so that the **weight of the cut is maximised.** It is proven to be **NP-HARD**

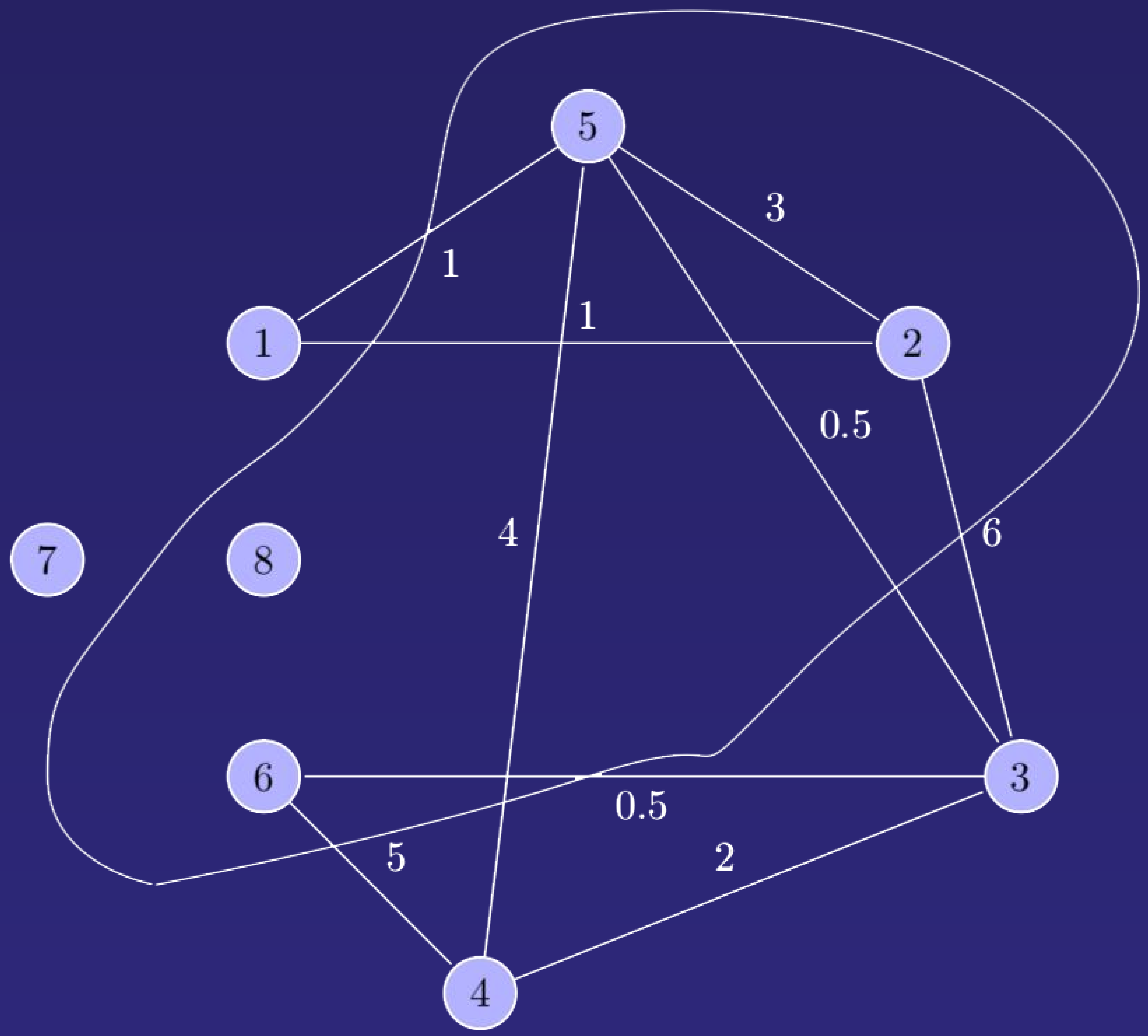
$$maximize\ Z = \frac{1}{2} \sum_{i \in V} \sum_{j \in I}^{i<j} w_{ij}(1 - x_i x_j)$$

$$S.T.\ \ x_i = \begin{cases} 1 & i \in S \\ -1 & i \in \bar{S} \end{cases}$$

$$w_{ij} \in \mathbb{R}^+$$

(Kartik & Karimi, 2007)

Max cut problem



In this case, the weight of the cut would be

$$0.5 + 6 + 1 + 1 + 5 + 0.5 = 14$$

And the cut's subset would be

$$C = \{2, 5, 6, 8\}$$

Max cut problem

Given a set of **n elements**, the goal is to find **a subset of m elements** in such a way that the sum of their distances **d_{ij} is maximized.**

(Kuo et al., 1993)

$$d_{ij} = \sqrt{\sum_{p=1}^{P} (s_{ip} - s_{jp})^2}$$

$$maximize\ Z = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j)$$

$$\sum_{i=1}^{n} x_i = m$$

$$x_i \in \{0, 1\}$$

$$i : i \in \mathbb{N} \wedge i \leq n$$

$$p : p \in \mathbb{N} \wedge p \leq P$$

Max diversity problem

$$H = \begin{bmatrix} h_{11} = 0 & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} = 0 & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & \cdots & h_{nn} = 0 \end{bmatrix} \wedge (h_{ij} \geq 0) \in \mathbb{R}^{n \times n}$$

Heuristics and greedy

Heuristics and greedy

Heuristic - from Greek **εὑρίσκω** "I find, discover".

“Exact algorithms might need centuries to manage with formidable challenges. In such cases heuristic algorithms that may find **approximate solutions** but have **acceptable** time and space **complexity** play indispensable role.”
(Kokash, 2018)

GREEDY ALGORITHM

RANDOM FIX

WEIGHTED LOCAL BRANCHING

Heuristics and greedy

G=(I,E) Graph from the set of players

H Adjacency matrix of weighted graph

1. **ALLOCATE SEEDS**
   In their respective clusters.
2. **COMPUTE WEIGHTED DEGREES**
   For each unseeded player.
3. **FOREACH i in I\{M}**
   1. **FOREACH k in K**
      if (cluster k is not full) &&
      (deltaZ<best_deltaZ)
      => $x_{ik}=1$

In order to create an initial feasible solution, the interpretation derived from the graph theory is fundamental. The greedy roots in the concept of **weighted degree.** Each player is assigned to the cluster in which the **increment of the O.F. is minimal.**

"Exploiting the specific structure of a given problem can provide a polynomial-time algorithm which achieves a quantitatively good O.F. "
(Della Croce et al., 2013)

Greedy

```java
int fixed_Q[] = new int[k];
int limit_Q = countQ / k;
for (int j = 0; j < k; j++) {
    bucket_heuristic[j] = new ArrayList<Integer>();
    for (int i = 0; i < n; i++) {
        if (isOne(z1[i][j]) && isOne(z2[i][j]) && players.get(i).getHas_seed() == false) {
            if (players.get(i).isnotQualified() == false && fixed_Q[j] < limit_Q) {
                fixed_Q[j]++;
                System.out.println("\t\tFixing a stable Q identified in X(" + i + "," + j + ")="
                        + (i + 1) * j + "=" + z1[i][j]);
                cplex.addEq(x[i][j], 1);
                z1[i][j] = 1.0;
                for (int jj = 0; jj < k; jj++) {
                    if (jj != j) {
                        cplex.addEq(x[i][jj], 0);
                        z1[i][jj] = 0.0;
                    }
                }
            } else {
                if (h_Degree[i][i] < H_max_degree * 0.55 && h_Degree[i][i] > H_max_degree * 0.45
                        && isOne(z1[i][j]))
                    bucket_heuristic[j].add(i);
            }
        }
    }
}

int randomIndex = -1;
int max_fixed = 0, randomVariable = 0;
for (int j = 0; j < k; j++) {
    max_fixed = (int) Math.ceil(bucket_heuristic[j].size() * 0.8);
    for (int i = 0; i < max_fixed; i++) {
        randomIndex = randomGenerator.nextInt(bucket_heuristic[j].size());

        }
        itcount++;
    }
```

Algorithm 1: Greedy algorithm for basic solution

Input: $\bar{e}, H, n, k, u$
Output: $\bar{x}$

$\Delta Z = \Delta\bar{Z} = 0$;
$best_j = 1; first = true$;
for $cluster = 1; cluster \leq j; cluster++$ **do**
  $Free_{cluster} = u$
end
foreach e_i *in* e **do**
  if $isSeeded(e_i) = false$ **then**
    $first = true$;
    for $cluster = 1; cluster \leq j; cluster++$ **do**
      $\Delta Z = 0$;
      if $Free_{cluster} > 0$ **then**
        for $player = 1; player \leq n; player++$ **do**
          if $x_{player,cluster} = 1$ **then**
            $\Delta Z += H_{player,e_i}$;
          end
        end
        if $first = true$ **then**
          $first = false$;
          $\Delta\bar{Z} = \Delta Z + 1$;
        end
        if $\Delta Z < \Delta\bar{Z}$ **then**
          $\Delta\bar{Z} = \Delta Z$;
          $best_j = cluster$;
        end
      end
    end
    $x_{e_i,best_j} = 1$;
    $Free_{best_j} --$;
  end
end

HeuB: Local branching

$$\sum^{n}\sum_{\bar{x_{ij}}=0}^{k} x_{ij} + \sum^{n}\sum_{\bar{x_{ij}}=1}^{k} (1 - x_{ij}) \leq k_{branch}$$

The amount of ***variables*** changing state is limited to the value $\mathbf{k_{branch}}$.

$$\sum^{n}\sum_{\bar{x_{ij}}=0}^{k} x_{ij} + \sum^{n}\sum_{\bar{x_{ij}}=1}^{k} (1 - x_{ij}) \geq k_{branch} + 1$$

If no improving solution is found, **move the search** to avoid cycling.

Inspired from the **Local Branching** (*Fischetti and Lodi, 2003*).

F&L LOCAL BRANCHING
Once an approximate solution is found, the solver - used as a ***black box*** - stops. A sub-problem is generated constraining the solver to search solutions in the **neighbourhood** of the **incumbent solution.** The process is **iterated**

WEIGHTED LOCAL BRANCHING
Some variables are **more weighty**. Therefore require more effort to be changed.

HeuA: Random fixing

$$\sum^{n} \sum^{k}_{\bar{x_{ij}}=0 \wedge i \in B} x_{ij} + \sum^{n} \sum^{k}_{\bar{x_{ij}}=1 \wedge i \in B} (1 - x_{ij}) + \sum^{n} \sum^{k}_{\bar{x_{ij}}=0 \wedge i \in A} \eta x_{ij} +$$

$$+ \sum^{n} \sum^{k}_{\bar{x_{ij}}=1 \wedge i \in A} \eta(1 - x_{ij}) \leq (\sum^{n}_{\bar{x_{ij}}=1 \wedge i \in A} \eta) k_{branch}$$

Algorithm 4: Heuristic b

$k_{branch\_init} = k_{branch}; constrain = null; counter = 0; bestOF = 0; ni - count$ $switch = true;$
while $ElapsedTime < T_l$ **do**
  $counter + +;$
  if $ni - count > ni - limit$ **then**
    $TL = -1$
  end
  Solver.Solve($timelimit = t_l;$);
  if $counter == 1$ **then**
    bestOF=Solver.getOF() +1;
  end
  if $Solver.gesStatus = FEASIBLE$ *or* $Solver.gesStatus = OPTIMAL$ **then**
    if $Solver.gesStatus = OPTIMAL$ *and* $noConstraint(\Delta Z(\bar{x}, x) \leq k_{branch}$ *or* $\Delta Z_w(\bar{x}, x) \leq k_{branch})$ **then**
      $TL = -1$/* Optimal found */
    end
    if $Solver.getOF() < bestOF$ **then**
      $ni - count = 0;$
      $\bar{x}$ = Solver.getSolution();
      Solver.startFrom($\bar{x}$);
      if $switch = true$ **then**
        Solver.setConstraint($\Delta Z_w(\bar{x}, x) \leq k_{branch} \cdot active$)/* active is the result from the sum in Eq. 4.3 */
      end
      else if $switch = false$ **then**
        Solver.setConstraint($\Delta Z(\bar{x}, x) \leq k_{branch}$);
        $switch = true;$/* local branching without weights */
      end
    end
    else
      $switch = false; ni - count + +; k_{branch} = k_{branch} \cdot \mu;$
      /* not improving */
      if $k_{branch} > n$ **then**
        $k_{branch}/2;$
      end
    end
  end
  else if $Solver.gesStatus = INFEASIBLE$ **then**
    $switch = false; k_{branch} = k_{branch} \cdot \mu;$
    Solver.removeConstraint($\Delta Z(\bar{x}, x) \leq k_{branch}$ and $\Delta Z_w(\bar{x}, x) \leq k_{branch}$);
    if $k_{branch} > n$ **then**
      $k_{branch}/2;$
    end
  end
end

20

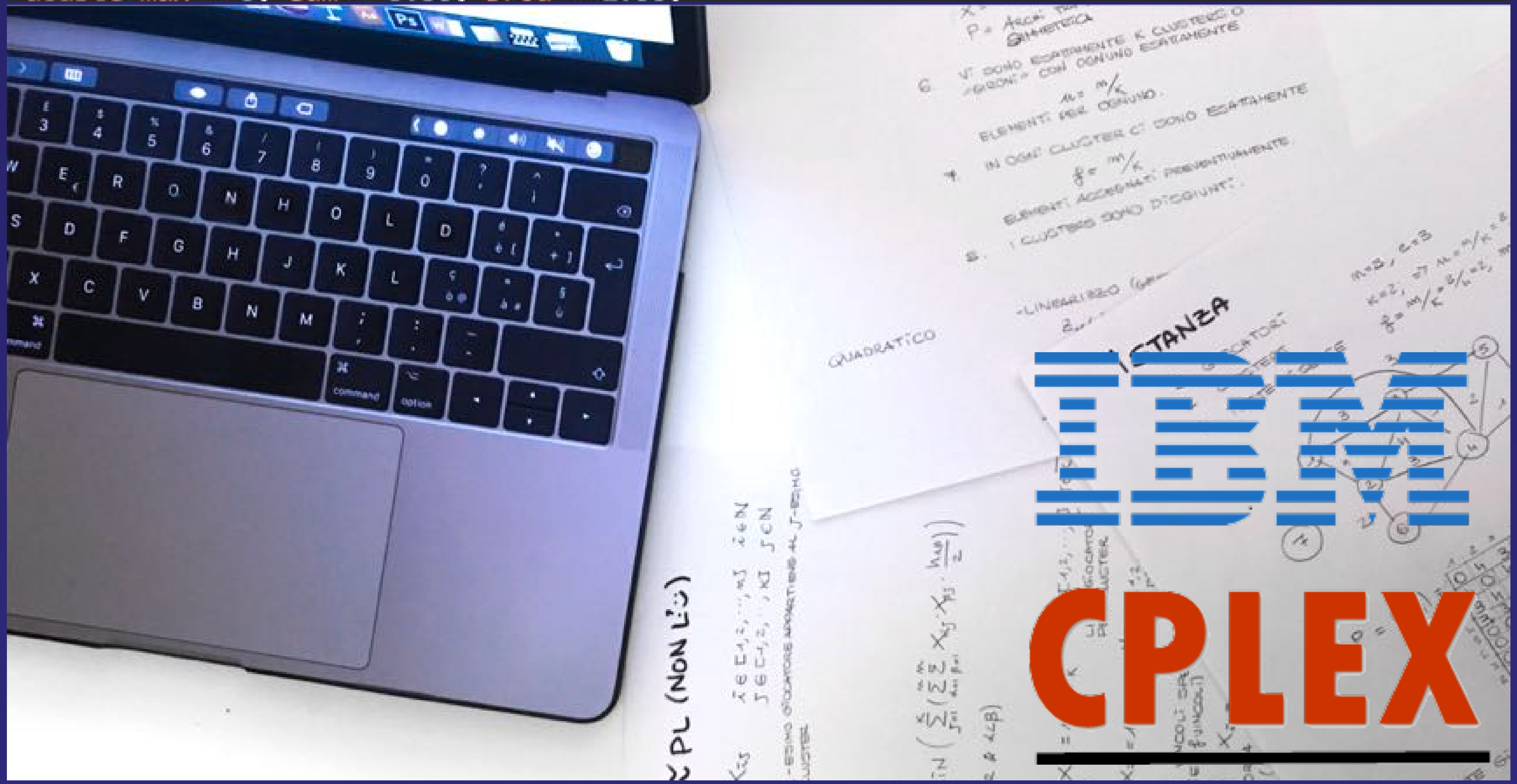
Simulations

Who does what?

The solver adopted is **IBM Ilog CPLEX,** commercial solver free for Academia.
The interface is implemented with **Java** and interacts with a **SQLite** database with players' data.

IBM ILOG CPLEX
SQLite DATABASE
JAVA

```
System.out.println("Checking properties of MatrixH:");
RealMatrix Mx = MatrixUtils.createRealMatrix(n, n);
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        Mx.setEntry(i, j, (Double) h[i][j]);
    }
}
EigenDecomposition testPSD = new EigenDecomposition(Mx);
System.out.println("\tHas complex eigen values: " + testPSD.hasComplexEigenvalues());
if (testPSD.hasComplexEigenvalues() == true) {
    throw new ErrorThrower("MatrixH is not semi-definite positive.Problem is not convex");
}
double[] eval = testPSD.getRealEigenvalues();
double max = 0, sum = 0.00, prod = 1.00;
```



Simulations flow chart

GENERATE MATRIX-H

TWO SOLUTIONS WITH **k=4**

THE GREEDY ONE | THE EXACT ONE

ITERATE DRAWS **u/2 TIMES**

DRAW 1st ROUNDS

SIMULATE MATCHES

GET A CHAMPION

$$P(\alpha, \beta) = 0.65 \cdot P_{rank}(\alpha, \beta) + 0.35 \cdot P_{H2H}(\alpha, \beta);$$

$$P_{H2H}(\alpha, \beta) = \frac{\#Wins\ of\ \alpha\ against\ \beta}{\#Matches\ of\ \alpha\ against\ \beta}$$

MatrixH rules

Given two generic players α and β, the coefficient $h_{\alpha\beta}$

+10	Come from the **same country**
+5	Played together in a **1st round** the last year.
+2	Played together in a **2nd round** the last year.
+1	Played together in a **3rd round** the last year.
+0.5	Played together in a **quarter or semi finals** the last year.

Tracked parameters

#	ACTUAL	CPLEX	GREEDY
LB	x	x	x
Optimal O.F. value	-	-	-
Solution status		x	x
O.F. value	x	x	x
O.F. % improvement		x	x
CPLEX O.F. with T_L=GreedyTime		x	
Greedy improvement with swaps			x
Number of active conflicts			
in the last generated first round	x	x	x
in the last whole tournament generated	x	x	x
Average number of active conflicts			
in every first round generated		x	x
in every whole tournament generated		x	x
Active conflicts measure			
in the last generated first round	x	x	x
in the last whole tournament generated	x	x	x
Average measure of active conflicts			
in every first round generated		x	x
every whole tournament generated		x	x

Table 5.2: Parameters for heuristic A

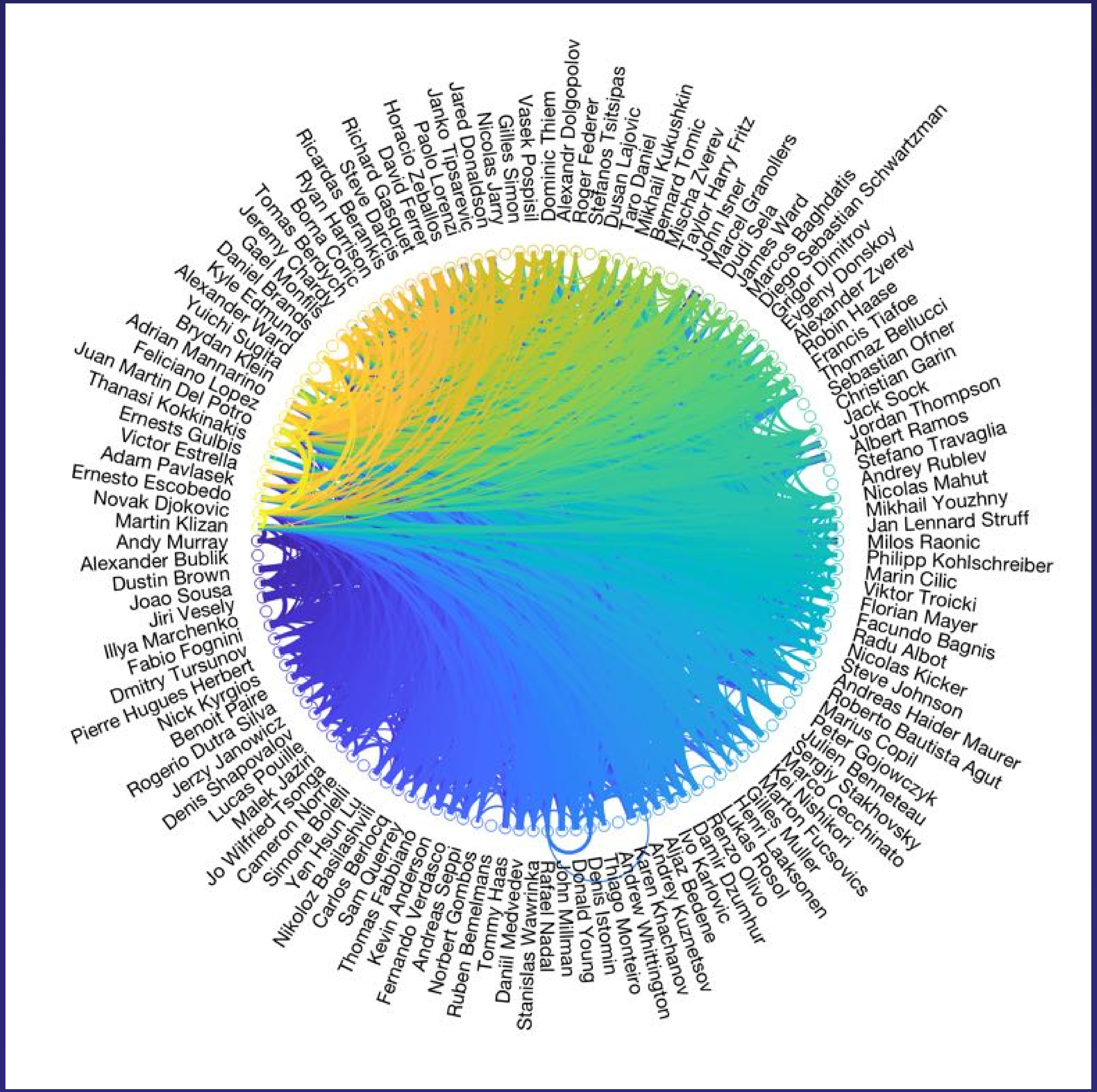
Parameter	Value
t_L	10s
θ	10
max_fixed	$bucket_j.size() \cdot 0.8$
λ_m	0.45
λ_M	0.55

Table 5.3: Parameters for heuristic B

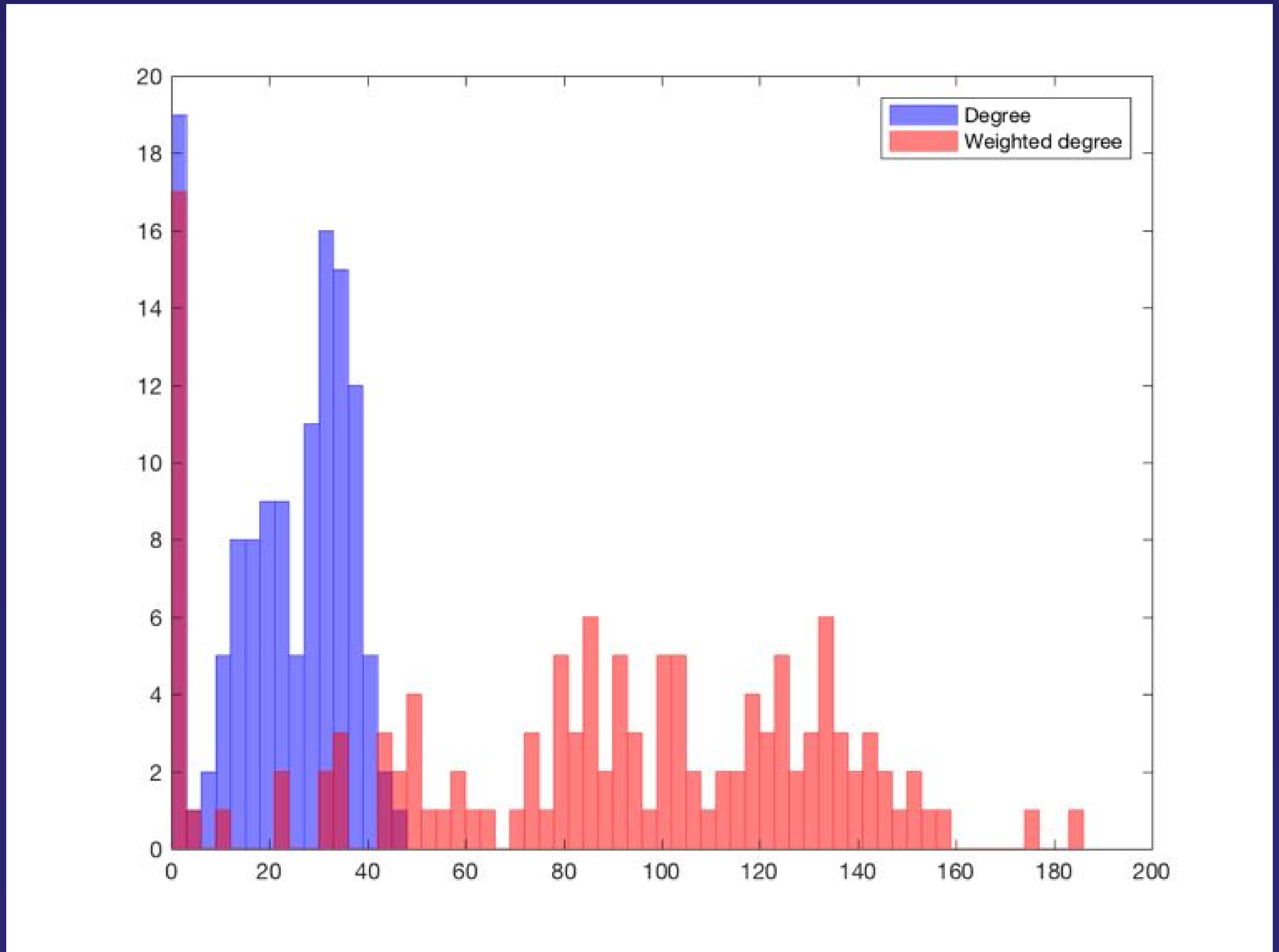
Parameter	Value
T_L	120s
t_L	7s
k_{branch}	$n \cdot 0.2 = 25$
μ	1.1
λ_m	0.35
λ_M	0.65
$ni - limit$	4

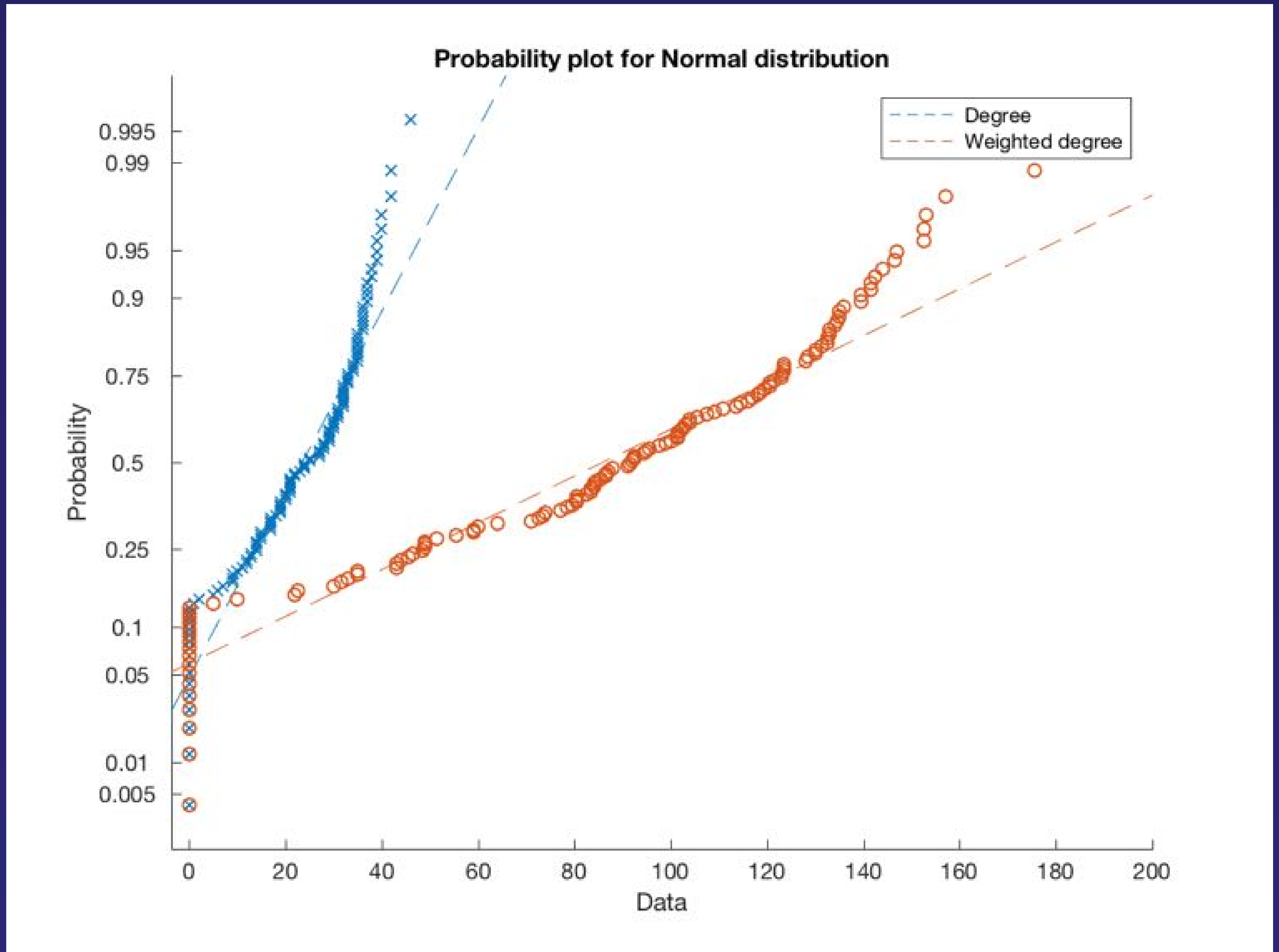
Results

Some graphics



Some graphics





Apparent fit of **weighted degrees** with the standard distribution.

Wimbledon results

	ACT	CPLEX	GREEDY
LB	162.5	162.5	162.5
Time	-	465.75s	1.88s
CPLEX O.F. with TL=GreedyTime	–	767.0	680.0
O.F.	1279.5	660.0 (48.42%)	680.0 (46.85%)
Greedy improvement with swaps	-	-	39.5
Solution status	-	Optimal	Feasible
Average indexes | s=16			
Number of conflicts in the 1st round	9	6.38	6.19
Measure of conflicts in the 1st round	28.5	16.56 (41.89%)	19.28 (32.35%)
Number of conflicts in the tournament	29	24.94	26
Measure of conflicts in the tournament	78.0	69.72 (10.62%)	71.62 (8.17%)
Last simulation indexes			
Number of conflicts in the 1st round	9	7	5
Measure of conflicts in the 1st round	28.5	20 (29.82%)	8 (71.93%)
Number of conflicts in the tournament	29	21	23
Measure of conflicts in the tournament	78.0	61.5 (21.15%)	62.5 (19.87%)

8min **vs** 2s.

O.F. 2.5% from optimality.

30% to 40% improvement in conflict measure. Avg. of 3 conflicts avoided.

Overall improvement in simulated tournaments.

Wimbledon results

Table 6.6: Winners for Wimbledon 2017 simulations

CPLEX	**GREEDY**
Andy Murray	Andy Murray
Stanislas Wawrinka	Andy Murray
Andy Murray	Jo Wilfried Tsonga
Stanislas Wawrinka	Andy Murray
Roberto Bautista Agut	Rafael Nadal
Stanislas Wawrinka	Rafael Nadal
Marin Cilic	Andy Murray
Andy Murray	Andy Murray
Marin Cilic	Andy Murray
Andy Murray	Marin Cilic
Lucas Pouille	Jo Wilfried Tsonga
Lucas Pouille	Roberto Bautista Agut
Marin Cilic	Rafael Nadal
Marin Cilic	Stanislas Wawrinka
Lucas Pouille	Lucas Pouille
Rafael Nadal	Ivo Karlovic

And the top players can still enjoy a good sleep :-)

Conclusions

Conclusions

FAIRER GAME

OPPORTUNITIES FOR PLAYERS

DIVERSITY AMONG MATCHES

PUBLIC ENJOY DIVERSITY

*And finally Tennis can join the super exclusive club of sports which benefit from Operations Research!

My drivers

I LIKE NUMBERS

I LIKE COMPUTERS

TERRIFIC MENTORS

CURIOSITY

https://github.com/gdragotto/TournamentAllocationProblem

Thanks

My tutors and mentors

Dr. Rosario Scatamacchia

Prof. Della Croce

Dr.Ing. Fabio Salassa

*For late-night biblical long emails, patience, support and encouragement during the whole time. And for the opportunities to come.

Kiitos

Which is *Thank you* in *Suomi*

Appendix

HeuA: Random fixing

1. **COMPUTE DEGREES**
   For each players
2. **COMPUTE θ SOLUTIONS**
   Using the solver as a black box with time t_l
3. **SELECT TWO BEST INCUMBENTS $x^{\theta 1}$ $x^{\theta 2}$**
   1. **FOREACH i in I\{M}**
      if (**is qualified**) &&
      ($x_{ij}^{\theta 1} = x_{ij}^{\theta 2} = 1$) &&
      (degree$_i$ in$[\lambda_m MD, \lambda_M MD] : \lambda_m, \lambda_M \in [0,1]$)
      => ADD IT TO A BUCKET
4. **EXTRACT FROM BUCKET**
   An η percentage of players and fix them in the sub problem **x_{ik}=1**
5. **COMPUTE SOLUTIONS**
   With 3/2 the initial time limit.

The approach leverage on a specific kind of player: **qualified players.** Since they are *newbies*, their **h-coefficients are null.**

This ***matheuristic*** (exact methods + heuristics) uses the solver as a **black box**. Once an **amount of solutions** is found - or the **TimeLimit** triggered- some qualified players are **fixed**, decreasing the DoF of the subproblem.
The process depends also on the **player's degree.**

HeuA: Random fixing


```java
int fixed_Q[] = new int[k];
int limit_Q = countQ / k;
for (int j = 0; j < k; j++) {
    bucket_heuristic[j] = new ArrayList<Integer>();
    for (int i = 0; i < n; i++) {
        if (isOne(z1[i][j]) && isOne(z2[i][j]) && players.get(i).getHa
            if (players.get(i).isnotQualified() == false && fixed_Q[j]
                fixed_Q[j]++;
                System.out.println("\t\tFixing a stable Q identified i
                        + (i + 1) * j + "=" + z1[i][j]);
                cplex.addEq(x[i][j], 1);
                z1[i][j] = 1.0;
                for (int jj = 0; jj < k; jj++) {
                    if (jj != j) {
                        cplex.addEq(x[i][jj], 0);
                        z1[i][jj] = 0.0;
                    }
                }
            } else {
                if (h_Degree[i][i] < H_max_degree * 0.55 && h_Degree[i
                        && isOne(z1[i][j]))
                    bucket_heuristic[j].add(i);
            }
        }
    }
}

int randomIndex = -1;
int max_fixed = 0, randomVariable = 0;
for (int j = 0; j < k; j++) {
    max_fixed = (int) Math.ceil(bucket_heuristic[j].size() * 0.8);
    for (int i = 0; i < max_fixed; i++) {
        randomIndex = randomGenerator.nextInt(bucket_heuristic[j].size
```

Algorithm 3: Heuristic A

```
1  Solver.Solve(timelimit = t_l; max_solutions = θ);
2  x^{θ1}=Solver.getSolution(best);
3  x^{θ2}=Solver.getSolution(best-1);
4  Allocated_k = 0;
5  for c = 1; c ≤ j; c++ do
6      for p = 1; p ≤ n; p++ do
7          if isSeeded(p) = false and x^{θ1}_{p,c} = x^{θ2}_{p,c} = 1 then
8              if isQualified(p) = true and Allocated_c < (Q_n/k);
9              then
10                 Solver.setconstraint(x_{p,c} = 1);
11             end
12             else if λ_m MD ≤ degree(p) ≤ λ_M MD then
13                 Bucket_c.add(x_{p,c});
14             end
15         end
16     end
17 end
18 for c = 1; c ≤ j; c++ do
19     max_fixed = Bucket_j.size() · η;
20     for rand = 1; rand ≤ max_fixed; rand++ do
21         fix=Bucket_c.getRandom();
22         Solver.setconstraint(fix=1);
23         Bucket_c.delete(fix);
24     end
25 end
26 Solver.StartFrom(x^{θ1});
27 Solver.Solve(timelimit = 3/2 · t_l;);
```