POLITECNICO DI TORINO

DIGEP - Department of Management and Production Engineering

Bachelor degree of Management and Engineering

Bachelor Degree Thesis

Fair players allocation in ATP Tournaments generation.

A combinatorial optimization approach.

Supervisors:
Prof. Federico Della Croce
Dr. Rosario Scatamacchia

Candidate
Gabriele Dragotto

Academic Year 2017-2018

Oh me! Oh life! of the questions of these recurring,
Of the endless trains of the faithless, of cities fill'd with the foolish,
Of myself forever reproaching myself, (for who more foolish than I, and who more faithless?)
Of eyes that vainly crave the light, of the objects mean, of the struggle ever renew'd,
Of the poor results of all, of the plodding and sordid crowds I see around me,
Of the empty and useless years of the rest, with the rest me intertwined,
The question, O me! so sad, recurring - What good amid these, O me, O life?

Answer.

That you are here - that life exists and identity,

That the powerful play goes on, and you may contribute a verse.

Walt Whitman

Acknowledgements

Using words properly, I think, is one of the most complex challenges we face in our lives. Especially when it comes to express sincere feelings to people we interacted with. How many different sets of words can we combine as to render the meaning of our intentions? It would be a great topic for Combinatorial Mathematics as well as for Behavioural Sciences. This can be the reason why I opted to work on an easier problem. After all, the solution space for the analyzed Tournament Allocation Problem is roughly made of $2^{(128-32)\cdot 4} \approx 10^{115}$ elements. Of course, that includes tons of infeasible solutions. But even a smaller number would still sound easier than finding good words as for ending this little journey.

> "[On NP-hard problems] These problems are hard because of the size of the haystack in which you're trying to find the needle... not the length of the description of the haystack." *Ryan Williams*

First of all, I have to thank my supervisors - Prof. Della Croce and Dr. Scatamacchia - which amazingly supported me during the time of this work. Since the beginning, they dedicated time and effort, and constantly made me enthusiastic as time was passing. They spent energies in reviewing late night emails and repeatedly pushed the limit forward. Questioning each assumption and hypothesizing about next developments. The pleasure I had working with them is one of the most interesting and pleasant outcomes I take back from my bachelor. Most of all, I have to recognize the extraordinary attitude - or value - they fostered in the process. **Curiosity**.

> "The creative force that moves one man to do mathematics is not unlike that force that moves another to write music or poetry, to produce art or literature." *David Sentilles*

Despite the feasible region is made of 8 *billions* potential solutions - hanging around somewhere in the globe - my closest friends are the optimal one. And I mean it not only for the support in recent times, in which I wrote formulas and draw graphs on their hankies, windows and - few times - papers. They bring - every single day - diversity, support, and affection into my life. They connect me with worlds I would not experience if not thanks to them. They create new edges between the nodes of my life, lighting up my mind and my heart. A special thought - or better a *Kiitos* - goes also to my dear friends from Finland - with which I shared the recent *Suomi* experience.

> "When two systems, of which we know the states by their respective representation, enter into a temporary physical interaction due to known forces between

them and when after a time of mutual influence the systems separate again, then they can no longer be described as before." *Erwin Schrödinger*

The trust my family always has in me deserves most of my gratefulness. My Father and my Mother - or better mom and dad - which are respectively the right and left hemisphere of my brain. My sister, which reminds me every time the beauty of being complementary. My dear aunt Rosalia, who raised me and always encouraged me to learn. Anna - my former Art teacher from high school - which taught me to appreciate the infinite through Arts. Each of you constantly gives me the strength to find my own verse.

"I have always believed, and I still believe, that whatever good or bad fortune may come our way we can always give it meaning and transform it into something of value." *Herman Hesse* in Siddhartha

The last credit I would like to give goes to the **curiosity**. It is no more than an ode to the unknown. To the countless questions of our lives. To the infinite possibilities of our existences. To the excitement of discovery but most for what we miss. To the ocean of the undiscovered. To the many stars we recognize in the skies, but most to the numberless ones we do not. To the answers we will not have. To the infinite mysteries hidden in the outwardly known. To the runs without clear destinations, the journeys without definite endings. To the curiosity which feels the wondrous breath of the unknown.

"Here, on the edge of what we know, in contact with the ocean of the unknown, shines the mystery and beauty of the world. And it's breathtaking." *Carlo Rovelli*

Abstract

Single-elimination tournaments are a popular type of tournament among sports, more specifically in tennis. Despite the current state-of-the-art procedures prevent seeded players - or highest rated opponents - from matching in early rounds, match repetitions among other players are possible, even in tournaments very close in terms of time. Therefore, the allocation process for non-seeded players plays a fundamental role in avoiding match repetitions and in increasing the diversity of matches.
The thesis develops a methodology for enforcing fairness in single-elimination tennis tournaments in terms of a reduction of match repetitions in consecutive different tournaments, without significantly altering the draw procedure. The considered tournament allocation problem amounts to solving a clustering problem by means of mathematical programming. Several results and solutions are provided for real-life instances related to Grand Slams in 2017 by exploiting the potential of an Integer Programming solver. Moreover, a greedy approach to generate quantitatively good solutions is presented, along with two heuristics. Full tournament simulations are performed to assess the quality of presented methodologies. Among the results, appreciable improvements are obtained for both the expected number of match repetitions and a related measure of fairness. Benchmarks between heuristics, the greedy algorithm, and the optimal solutions are presented. An important outcome is related to the quality of solutions built with the greedy algorithm. Some techniques of data visualization are implemented to highlight the obtained results.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and drivers

1.1 Single-elimination and fairness

The single-elimination - or sudden-death - 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. The tournament ends when a final match between the last two players results in a winner, which is the tournament champion. Sudden-death is a common type of elimination tournament in sports such as football, tennis, and baseball.

This type of tournament has been deeply studied in Statistics and Combinatorial Mathematics. Horen & Riezman (1985) review several studies on the matter, pointing out how different configurations for a generic tournament lead to diverse patterns of winners and losers. Therefore, according to their paper, the tournament configuration might favor or disadvantage contenders. Furthermore, Williams (2010) has shown that - under certain assumptions - there is always a specific tournament structure which maximizes the odds of winning for any generic player. Moreover, according to the authors, a single-elimination tournament might be won by any player which exceed a determined amount of victories in an equivalent round robin tournament. Hence, because of several players might be potential champions for the same tournament, some concerns about fairness and the final ranking arise.

Therefore, dealing with fairness on a macroscopic scale can turn out to be very complex. This thesis focuses on **fairness in terms of match repetitions**, trying to limit repeated matches during a given period of time. Moreover, *parameters such as surface type, country, and entry level of players are taken into account* as potential elements of disparity and conflict between players. Clearly, the proposed analysis can be as well extended to any other parameter of interest.

1.2 Match repetitions

There are several cases in which players have been paired with the same opponent multiple times during a time-window ranging from 1 to 3 months. This implies a decreased diversity of matches among players, which might make games **less interesting for the public** as well as for the players. Empirical evidence from ATP supports those claims.
As a matter of facts, the ATP database provided by Sackmann (2017) - the creator of TennisAbstract.com - provides several examples of match repetition. In particular, we focused on the data regarding first rounds of 128, 64, and 32 players tournaments held in 2017. The Table 1.1 reports match repetitions for players in first rounds of several ATP tournaments from January to September 2017. In addition to that, there are few cases in which - as reported by Sackmann (2018) - finalists in a given tournament play against each other in first rounds of the next tournament. For instance, Diego Schwartzman and Fernando Verdasco played against each other in the final round of Rio de Janeiro ATP-500, and again in the first round of Acapulco the following week. While the last case is definitely rare, match repetitions in early rounds are very common in ATP tournaments.

1.3 Variety of the matches and randomness of the draw

Variety - or diversity - is the extent by which different elements of a set differ by one or more characteristics. In the specific application of this work, ensuring a reasonable variety means trying to create the more diverse match pairings in tournaments. Hence, the aim of avoiding match repetitions fits perfectly with the concept of variety. Moreover, this goal can be achieved by integrating several parameters to measure diversity, such as nationality of players, performances in a given period of time, number of matches played.
One of the fundamental assumptions of this thesis is related to 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.

Table 1.1: Match repetitions from January to September 2017 in ATP Grand Slams and Challengers. Adapted from Sackmann (2017) database.

Tourney Name	Size	Month	Winner	Seed	Loser	Seed	W/L
Qingdao CH	32	April	Andrej Martin	7	Aleksandr Nedovyesov	NO	
Prague CH	32	July	Andrej Martin	4	Aleksandr Nedovyesov	NO	
							2/0
Sophia Antipolis CH	32	April	Arthur De Greef	8	Vincent Millot	NO	
Bordeaux CH	32	May	Arthur De Greef	NO	Vincent Millot	NO	
							2/0
Busan CH	32	May	Blaz Kavcic	3	Sekou Bangoura	NO	
Granby CH	32	July	Blaz Kavcic	1	Sekou Bangoura	NO	
							2/0
Shenzhen CH	32	March	Duck Hee Lee	4	Tatsuma Ito	NO	
Gimcheon CH	32	May	Duck Hee Lee	7	Tatsuma Ito	NO	
							2/0
Australian Open	128	January	Dudi Sela	NO	Marcel Granollers	NO	
Wimbledon	128	July	Dudi Sela	NO	Marcel Granollers	NO	
							2/0
Leon CH	32	March	Federico Coria	NO	Tigre Hank	NO	
San Luis Potosi CH	32	April	Federico Coria	NO	Tigre Hank	NO	
							2/0
Yokohama CH	32	February	Go Soeda	4	Hiroki Moriya	NO	
Taipei CH	32	April	Go Soeda	8	Hiroki Moriya	NO	
							2/0
Santiago CH	32	March	Guilherme Clezar	NO	Tristan Lamasine	NO	
Liberec CH	32	July	Guilherme Clezar	NO	Tristan Lamasine	NO	
							2/0
Wroclaw CH	32	February	Jonathan Eysseric	NO	Hubert Hurkacz	NO	
Poznan CH	32	July	Jonathan Eysseric	NO	Hubert Hurkacz	NO	
							2/0
Medellin CH	32	July	Juan Pablo Varillas Patino Samudio	NO	Facundo Mena	NO	
Floridablanca CH	32	August	Juan Pablo Varillas Patino Samudio	NO	Facundo Mena	NO	
							2/0
Sophia Antipolis CH	32	April	Kimmer Coppejans	NO	Stefanos Tsitsipas	NO	
Barletta CH	32	April	Kimmer Coppejans	NO	Stefanos Tsitsipas	NO	
							2/0
Granby CH	32	July	Liam Broady	NO	Marc Polmans	NO	
Vancouver CH	32	August	Liam Broady	NO	Marc Polmans	NO	
							2/0
Barletta CH	32	April	Matteo Donati	NO	Andrea Pellegrino	NO	
Genova CH	32	September	Matteo Donati	NO	Andrea Pellegrino	NO	
							2/0
Prostejov CH	32	June	Radu Albot	6	Facundo Bagnis	NO	
Wimbledon	128	July	Radu Albot	NO	Facundo Bagnis	NO	
							2/0
Leon CH	32	March	Tennys Sandgren	6	Mackenzie Mcdonald	NO	
Savannah CH	32	May	Tennys Sandgren	5	Mackenzie Mcdonald	NO	
							2/0
Quanzhou CH	32	March	Yuki Bhambri	NO	Tatsuma Ito	NO	
Gatineau CH	32	July	Yuki Bhambri	NO	Tatsuma Ito	NO	

Chapter 2

Definitions and formulation

2.1 Tennis sudden-death

In the specific field of Tennis, single-elimination tournaments are the standard for ATP and WTA competitions. In the majority of ATP 500,1000 and Grand Slam tournaments, the number of players is either 32, 64 or 128. A common practice is *seeding*, by which a set of players are allocated in some specific slots for the first round. Following a preliminary ranking, those players are positioned into the tournament brackets graph in advance, before the general draw for unseeded players. Most of the times, the set of seeded players includes the best performing players in the tournament. The allocation of seeds is required in order to ensure that top players do not eventually match each other until late or final rounds.

Assuming n players are joining a sudden death tournament, the number of matches is given by $n-1$. The amount of players n is always a power of two, namely $n = 2^t$ for some positive number t. The tournament structure represents the way players are facing each other in the tournament. Graphically, the structure is represented by a brackets graph such as the one in Figure 2.1

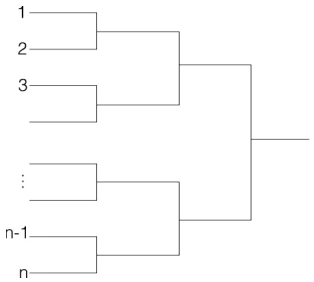


Figure 2.1: Brackets graph

In order to create the first round of a tournament, each player has to be assigned to a specific slot in the brackets graph. As an additional requirement for the model, we suppose that tournaments have *no byes*. A bye occurs when a player or a team does not play in a generic round, playing directly in the following one.

2.2 Seeded players

In the majority of tennis tournaments, best players are seeded typically following the WTA (Women Tennis Association) or ATP (Association of Tennis Professionals) rankings. Several tournaments compute rankings as to reflect a particular characteristic of a competition: for instance, the Wimbledon seeding methodology favors players which are better performing on grass rather than other surfaces (Wimbledon 2017).
Once the **seeds ranking** is set up, seeded players are allocated into the brackets graph. Only the first two seeds usually have an *a priori* allocated slot, while the remaining seeds have some specific slots in which they can be allocated. Since more than one seed might be allocated to the same slot, a draw is made. The slot location in the brackets graph is given by the round in which players are expected to eventually match each other. For instance, in a Grand Slam with $n = 128$ players and $m = 32$ seeds:

1. The top 2 players are allocated in the first spot and the last one, so that - assuming they will win all their games - they play against each other in the final round.

2. The top 4 players eventually play against each other in the semi-finals

3. The top 8 players eventually play against each other in the quarter-finals, the top 16 in the round before quarters and so on.

For the purpose of this work, we assume the *seeding is given*. Hence, a set of seeded players and their positions in the first round is provided.

2.3 Problem definition

Let us first provide the following definitions.

Definition 1 A tournament is a single-elimination tournament with no byes and an amount of players $n = 2^t : t \in \mathbb{N}$.

Definition 2 A round is a set of matches between z players which results in $z/2$ winners. In a generic tournament, the number of rounds is $rounds = log_2 n$. The set of rounds in the tournament is defined as $R = \{r_i \in \mathbb{Z} : 0 \leq r_i < t\}$

Definition 3 $I = \{i \in \mathbb{N} : 1 \leq i \leq n\}$ is the set of players in the tournament.

Definition 4 The structure of a tournament is a mapping that assigns each player $i \in I$ to a single slot in $S = \{s_i \in \mathbb{N} : 1 \leq s_i \leq 2(n-1)\}$

In the first round - or $r_i = 0$ - for a tournament of n players there are exactly $n/2$ matches and n slots. Therefore, each slot can be indexed with a unique number ranging from 1 to n. The next round has $n/2$ players with $n/4$ matches, and indexes ranging from $n+1$ to $n+1+n/2$.

Definition 5 A generic round r_i has slot numbers which can be indexed as follow: $s_i \in [1 + \sum_{l=0 \wedge r_i > 0}^{r_i - 1} 2^{rounds-l}, \sum_{l=0}^{r_i} 2^{rounds-l}]$

Definition 6 A match repetition occurs when two players play against each other for more than one time in a given time window.

With those definitions, the problem can be formulated as follow.

Definition 7 The **tournament allocation problem** clusters n players with m seeds in k groups, in order to *maximize diversity and minimize match repetitions.* Matches within clusters are then randomly determined by a draw.

As noted in Sections 1.2 and 1.3, several parameters are potential candidates as to maximize some sort of diversity, as long as a randomized draw takes place. Hence, the problem is related to a process which **splits players into different clusters** - or groups. The clustering minimizes some sort of numerical measure for match repetitions and maximizes diversity in terms of a set of given parameters. Once clusters are created, the draw can randomly assign non-seeded players to their position in the brackets graph. Therefore, the solution of the tournament allocation problem does not provide certainty - in terms avoided match repetitions and increased diversity - but rather increases the odds of creating such a tournament structure. As mentioned in Section 2.2, some players are fixed in their specific clusters because of their seeding position. Those seeded players reduce the degrees of freedom for the problem, hence making it easier to solve.

Definition 8 $k : k \in \mathbb{N} \wedge \ (n \ mod \ k) = 0$ is the number of clusters - or disjoint sets of players - in a tournament structure. Therefore, each cluster has an amount of $u = \frac{n}{k} \in \mathbb{N}$ players. $J = \{j \in \mathbb{N} : 1 \leq j \leq k\}$ is the set of clusters

The Definition 8 assumes that the number of cluster allows the creation of subsets with the same cardinality.

Definition 9 A seeded player is a player which has an *a priori* assigned slot in the brackets graph. $M \in I$ is the sub-set of seeded players in a tournament, with cardinality $m < n$. Each cluster has an amount of $f = \frac{m}{k} \in \mathbb{N}$ seeded players.

Moreover, some players join the tournament following a qualification process. Hence, newly qualified players need to be taken into account and treated as special entities. Different types of players can be classified as follow:

1. Seeded players
2. Unseeded players
   (a) *Qualified players* Q are players which join the tournament from some sort of qualification phase. Hence, the amount of match repetitions with other players is arbitrarily assumed to be 0. This set includes lucky losers LL - the bests non-qualified players in the qualification phase - which join the tournament when a competitor is not able to play.

(b) *Non-qualified unseeded players*

We introduce a matrix H for representing the cost of a match repetition between two players as well as other costs related to conflicts.

Definition 10 The matrix H is a matrix $n \times n$ in which the generic element $h_{\alpha\beta} \in \mathbb{R} \wedge h_{\alpha,\beta} \geq 0$ represents some sort of interaction between two generic players $\alpha, \beta : \alpha, \beta \in I$

Hence, h coefficients quantify some sort of conflict or interaction between two generic players, which is to be minimized. For the problem purpose, we assume that $h_{\alpha\beta} \in R : h_{\alpha\beta} \geq 0$. Moreover, the matrix H is symmetric and has a diagonal of zeros. The former assumption roots in the rules defined in order to create h coefficients. In particular, we assume that $h_{i,i} = 0 \; \forall i \in I$ and $h_{\alpha,\beta} = h_{\beta,\alpha} \; \forall \alpha, \beta \in I$.

$$H = \begin{bmatrix} h_{11} = 0 & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} = 0 & \cdots & h_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ h_{n1} & h_{n2} & \cdots & h_{nn} = 0 \end{bmatrix} = (h_{ij} \geq 0) \in \mathbb{R}^{n \times n} \tag{2.1}$$

Definition 11 A conflict between to players $\alpha, \beta \in I$ exists when one or more incompatibility is detected between them.

For instance, if we define match repetitions as a type of incompatibility, a conflict is detected each time two players have played against each other in the past. The former example can be restricted, for example, to the current season, so that only recent match repetitions are taken into account. Following the last definition, it is evident how *conflicts* are intrinsically linked to coefficients in the matrix H. Moreover, we can distinguish between potential conflicts and activated one.

Definition 12 Two generic players $\alpha, \beta \in I$ have a potential conflict if $h_{\alpha,\beta} > 0$.

Definition 13 A conflict is activated when two generic players $\alpha, \beta \in I$ are playing against each other in a match and $h_{\alpha,\beta} > 0$. Hence, the conflict measure is defined as $h_{\alpha,\beta}$.

Therefore, a generic conflict between two players becomes *active* only if the two are actually allocated in slots as to play together and their $h_{\alpha,\beta}$ is positive.

2.4 Integer Quadratic Programming formulation

The tournament allocation problem can be stated in terms of a quadratic 0/1 optimization problem. As referenced in Section 2.3, the goal is to create clusters of players as to minimize some sort of *conflicts* - for instance, match repetitions - and increase *diversity* across matches. In particular, the goal is to optimize the feasible draws related to the **first round**.

$$minimize\ Z = \sum_{j=1}^{k}(\sum_{\alpha=1}^{n-1}\sum_{\beta=\alpha+1}^{n} h_{\alpha\beta}x_{\alpha j}x_{\beta j}) \tag{2.2}$$

S.T.

$$\sum_{j=1}^{k} x_{ij} = 1 \ \ \forall i \in I \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ij} = u \ \ \forall j \in J \tag{2.4}$$

$$x_{ij} \in \{0,1\} \ \ \forall i \in I, j \in J \tag{2.5}$$

$$\alpha, \beta, i \in I, \ j \in J \tag{2.6}$$

The generic variable x_{ij} is equal to 1 each time the player i is allocated to the cluster j.
The O.F. in Equation 2.2 minimizes the sum of conflicts for a generic cluster $j \in J$. Hence, if two players $\alpha, \beta \in I$ are both allocated to the same $j \in J$, the product $h_{\alpha\beta}x_{\alpha j}x_{\beta j}$ becomes active and assumes the value of $h_{\alpha\beta}$. In any other case, this contribution is equal to 0. In order to obtain the objective function, a sum through every cluster is performed. As reported in Section 2.3, matrix H is symmetric and has a zero-diagonal. Therefore, as to simplify the O.F., only inner sums so that $\alpha < \beta$ are taken into account. A more formal implementation is given in Equation 2.2, which uses indexes on the top and bottom of the sums as to obtain the desired effect.
The constraint in Equation 2.3 ensures that each player is allocated to one and only one cluster, while Equation 2.4 sets the maximum amount of players per cluster to u.

A generic seeded player $i \in M$ is *a priori* positioned in a slot $s_i \in S$ in the first round. Therefore, an amount of f variables for each cluster j are fixed, so that their binary variables x_{ij} are set to 1.

2.5 Integer Linear Programming formulation

We can derive an alternative Integer Linear Programming model by applying the standard linearization to the product of binary variables, as mentioned in Della Croce (n.d.). Indeed, the linearization of a generic product $x_i x_j$ can be achieved by introducing a new binary

variable $y_{ij} \in \{0,1\}$ such that $y_{ij} = x_i x_j$. In addition to that, 3 new constraints are declared:

$$\begin{aligned} x_i &\geq y_{ij} \\ x_j &\geq y_{ij} \\ x_i + x_j &\leq 1 + y_{ij} \end{aligned} \tag{2.7}$$

In particular, as to linearize the O.F.in Equation 2.2, $y_{\alpha j,\beta j} = x_{\alpha j} x_{\beta j}$ is introduced. Therefore, 3 new constraints are declared in Equation 2.9, and a new O.F. is introduced in 2.8.

$$minimize\ Z = \sum_{j=1}^{k} (\sum_{\alpha=1}^{n-1} \sum_{\beta=\alpha+1}^{n} h_{\alpha\beta} y_{\alpha j,\beta j}) \tag{2.8}$$

S.T.

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

$$\begin{aligned} x_{\alpha j} &\geq y_{\alpha j,\beta j} \\ x_{\beta j} &\geq y_{\alpha j,\beta j} \\ x_{\alpha j} + x_{\beta j} &\leq y_{\alpha j,\beta j} + 1 \\ y_{\alpha j,\beta j} &\in \{0,1\}\ \forall i \in I, j \in J \end{aligned} \tag{2.9}$$

$$\begin{aligned} x_{ij} &= 1\ \forall i \in M \\ x_{ij} &\in \{0,1\} \ \ \forall i \in I, j \in J \\ \alpha, \beta, i &\in I,\ j \in J \end{aligned}$$

The new model might achieve better performances on solvers which do not support the automatic linearization of the O.F. or are not able to process quadratic O.F. As reported in Chapter 6, the solver used in the simulations for this thesis automatically linearizes the O.F.

Chapter 3

Related problems

The tournament allocation problem introduced in Section 2.4 is a particular formulation which can be lead back to the broad set of clustering problems. Therefore, there are several similar problems in the literature. As to be more specific, two problems are presented in Sections 3.1 and 3.2. Those problems have similar characteristics if compared to the tournament allocation problem.

3.1 Max-cut problem

The tournament allocation problem shares several characteristics with an undirected and weighted max-cut problem. Each player $i \in I$ is represented by a vertex in a graph, and the given matrix H is interpreted as the *adjacency matrix*. The greater the weight of an edge is, the more the two linked nodes have some sort of *conflict*.

For instance, in the example graph in Figure 3.1, is very immediate to visualize that players 7 and 8 have no connections with other players. Hence, the cluster in which they virtually can be allocated is not quantitatively relevant for the O.F. Moreover, players 2 and 3 have a weighty edge which may cause a sensible increase in the O.F. - in the case they are allocated in the same cluster. Hence, the problem of clustering players can be visualized as a *closed line* which creates a subset of nodes. In order to achieve a good clustering in terms of *diversity and match repetitions*, the line has to maximize the sum of edges' weights which are cut. In Figure 3.1, the red line clusters a subset $C = \{2,5,6,8\}$ and has a cut-value of $0.5 + 6 + 1 + 1 + 5 + 0.5 = 14$. As to transpose this methodology to the tournament allocation problem, k lines are required to create k disjoint clusters. In terms of complexity, the *max-cut problem* is proven to be **NP-Hard** (Kartik & Karimi 2007). The graph for a tournament of 32, 64 or 128 players is harder to visualize but can be helpful as to represent the tournament allocation problem. An example of a full tournament graph is reported later on in Figure 6.4.

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 subsets $S, \bar{S} : S + \bar{S} = V \wedge S \bigcap \bar{S} = \emptyset$. The *weight of the cut* is defined as the sum of edges' weights which have endpoints in the two

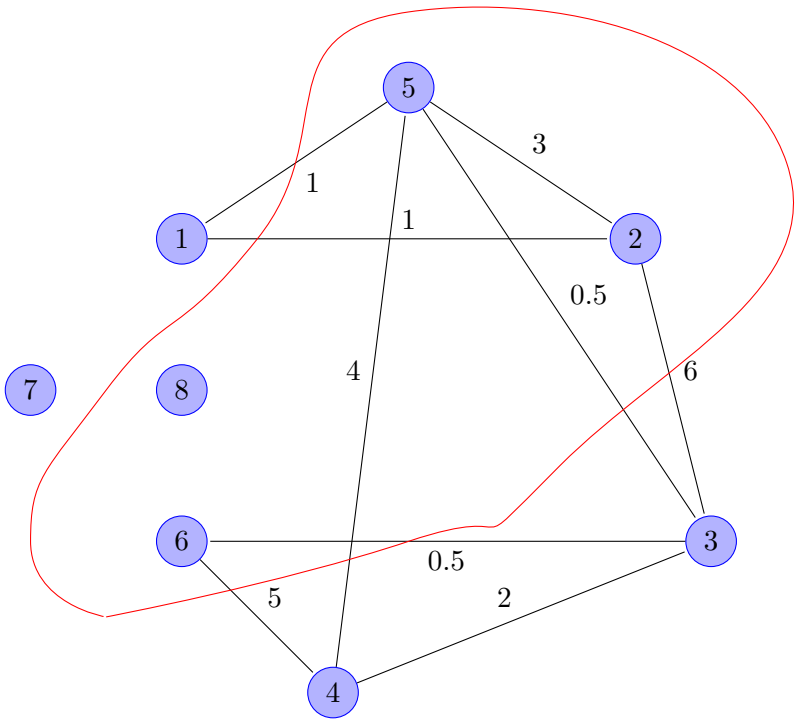


Figure 3.1: Example graph with $n = 8$ players

different subsets. x_i is introduced for each vertex $i \in V$ of $G(V, E)$. For each edge between two generic nodes $i, j \in V$, w_{ij} represents its weight. One of the many formulations for the max-cut problem is the one reported in Equations 3.1.

$$\begin{aligned} maximize\ Z &= \frac{1}{2} \sum_{i \in V} \sum_{j \in I}^{i<j} w_{ij}(1 - x_i x_j) \\ S.T.\ \ x_i &= \begin{cases} 1 & i \in S \\ -1 & i \in \bar{S} \end{cases} \\ w_{ij} &\in \mathbb{R}^+ \end{aligned} \tag{3.1}$$

The term $(1 - x_i x_j)$ assumes the value of 0 if the two nodes i, j belong to the same subset. In any other case, the term is equal to 2, and then compensated by the $1/2$ factor before the sum. Max-cut resembles the tournament allocation problem with $k = 2$ clusters. Nodes represent players and edges the quantification - or measures - of conflicts. In order to obtain the problem with k clusters, we need to generalize the max-cut. Therefore, the formulation introduced in Section 2.4 is very similar to a multiple max-cut.

3.2 Max-diversity problem

The tournament allocation problem has some characteristics in common with the *max-diversity 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.

$$\begin{aligned} maximize\ Z &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j) \\ d_{ij} &= \sqrt{\sum_{p=1}^{P} (s_{ip} - s_{jp})^2} \\ \sum_{i=1}^{n} x_i &= m \\ x_i &\in \{0,1\} \\ i : i &\in \mathbb{N} \wedge i \leq n \\ p : p &\in \mathbb{N} \wedge p \leq P \end{aligned} \tag{3.2}$$

Assuming s_{ip} as the value of the p^{th} attribute for the generic element i, the O.F. maximizes the sum over each p attribute of the set. Kuo et al. (1993) propose the formulation such as the one reported in Equation3.2. The concept of *distance* is dynamical and can be fit into several different problems. For instance, we can define some sort of distance so that match repetitions pull players closer. Hence, the max diversity tries to maximize the distance between players, avoiding match repetitions. In terms of computational complexity, the Authors prove that the max-diversity problem is **NP-hard**. By extending the max-diversity to k clusters and defining distance d_{ij} in a way that fits with Section 2.3, the problem resembles the tournament allocation problem.

Chapter 4

Heuristics and greedy

4.1 A greedy algorithm

In order to create an initial feasible solution, the interpretation derived from the graph theory is fundamental. As mentioned in Section 3.1, the tournament allocation problem can be represented through a graph $G = (V, E)$ where $V = I$ is the set of players and H is the adjacency matrix for the graph. The degree of a node $i \in I$ is defined as the number of edges incident to i. Hence, given the adjacency matrix H, the degree of a generic vertex i can be obtained counting how many non-zero values are present either on a i^{th} row or column. The degree of a node $i \in I$ is equivalent to the number of players with which i has at least one conflict. Since $h_{\alpha,\beta} \geq 0 \ \forall \alpha, \beta \in I$, summing the weights of each edge incident to a generic node i results in its *weighted degree*. Therefore, the weight of each edge measures the conflict value between two nodes. Nodes with a large weighted degree are the ones with the greatest amounts of potential conflicts.

The distribution of degrees and weighted degrees, as the ones reported in Figure 6.1, seems to follow almost a normal distribution. Hence, a methodology to create an initial solution has been developed. A feasible solution can be generated by prioritizing the allocation of players with higher weighted degrees. Seeded players and their clusters are still considered as fixed, as in Section 2.4. Starting with the player with the greatest $degree_w$, players are allocated in clusters. Each player $i \in I$ is taken in account individually, and virtually allocated in a cluster $j \in J$. More precisely, the algorithm starts when only seeds are already positioned in their clusters. In the second instance, the change Δz in the O.F. is computed by virtually allocating player i in the j^{th} cluster. The process iterates k times for each $j \in J$, and i is allocated to the cluster with the smallest and best $\Delta \bar{z_j}$ and at least 1 free space available. The vector $\bar{e}$ contains the ordered set of $i \in I$ by their weighted degrees, so that $degree_w(\bar{e}[a]) \geq degree_w(\bar{e}[a+1]) \ \forall a \in \mathbb{N} \ \wedge 1 \leq a \leq (u \cdot k)$. The process iterates through $\bar{e}$ and is represented in Algorithm 1. Therefore, players are allocated starting with the ones with higher weighted degrees. After every iteration, the next one takes into account all the previous allocations.

Algorithm 1: Greedy algorithm for basic solution

```
Input: ē, H, n, k, u
Output: x̄
ΔZ = ΔZ̄ = 0;
best_j = 1; first = true;
for cluster = 1; cluster ≤ 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 ≤ j; cluster++ do
            ΔZ = 0;
            if Free_cluster > 0 then
                for player = 1; player ≤ n; player++ do
                    if x_player,cluster = 1 then
                        ΔZ+ = H_player,e_i;
                    end
                end
                if first = true then
                    first = false;
                    ΔZ̄ = ΔZ + 1;
                end
                if ΔZ < ΔZ̄ then
                    ΔZ̄ = ΔZ;
                    best_j = cluster;
                end
            end
        end
        x_e_i,best_j = 1;
        Free_best_j − −;
    end
end
```

Once a *basic* feasible solution is generated, as previously described, a **random swapping procedure** is launched for $swap_t$ seconds. In the first instance, the procedure creates a bucket containing all the unseeded players in the set $B = I - M$ with cardinality $u \cdot k$. Then, two players $\alpha, \beta \in B$ are randomly extracted from the bucket. Assuming that, at the beginning of the iteration, α is allocated in $j_\alpha \in J$ and β in $j_\beta \in J$ with $j_\alpha \neq j_\beta$, the two players are virtually swapped in terms of clusters. Hence, Z_{α,j_α} and Z_{β,j_β} are defined as the contributions to the O.F. with the two players allocated in their original clusters. Then, the algorithm computes the two contributions to the O.F. in a virtual case of a swap

Z_{α,j_β} and Z_{β,j_α}. If the quantity $\Delta Z = (Z_{\alpha,j_\beta} + Z_{\beta,j_\alpha}) - (Z_{\alpha,j_\alpha} + Z_{\beta,j_\beta})$ is negative, the swap is improving the O.F. Therefore, players are swapped in the solution. The procedure is reproduced in Algorithm 2.

Algorithm 2: Swapping procedure

Input: $\bar{x}, H, n, swap_t$
Output: $\bar{x}$

```
1  while Time<swap_t do
2      while j_α = j_β do
3          x̄_{α,j_α}–>random(in B); x̄_{β,j_β}–>random(in B);
4      end
5      for i = 1; i ≤ n; i++ do
6          if x̄_{i,j_β}=1 then
7              if i ≠ β then
8                  Z_{α,j_β}+=H_{α,i};
9              end
10             Z_{β,j_β}+=H_{β,i};
11         end
12         if x̄_{i,j_α}=1 then
13             if i ≠ α then
14                 Z_{β,j_α}+=H_{β,i};
15             end
16             Z_{α,j_α}+=H_{α,i};
17         end
18     end
19     if ΔZ < 0 then
20         x̄_{α,j_α} = 0; x̄_{α,j_β} = 1;
21         x̄_{β,j_β} = 0; x̄_{β,j_α} = 1;
22     end
23 end
24 return x̄
```

4.2 Heuristics and matheuristics

Heuristics provide feasible solutions which might become available in a reasonable amount of time and achieve a good approximation for the optimal O.F. Because of the majority of solvers have several heuristics and exploration policies implemented, outperforming them implies exploiting some peculiar characteristics for each specific problem.
As noted by Della Croce et al. (2013), exploiting the specific structure of a given problem can provide a polynomial-time algorithm which achieves a quantitatively good O.F. Most of the times, the application of heuristics results in feasible solutions, while in few cases it leads to the optimal value. The principle behind *matheuristics* is the hybridization of exact-methods with heuristics, combining the strengths of the two. In several applications of matheuristics, the solver is used as a *blackbox* for multiple instances, in order to explore the solution space in different ways. Two approaches are presented in the following subsections, while results are analyzed in Chapter 6.

4.2.1 Heuristic A - Random fixing

The following approach tries to leverage on a specific type of players. Qualified players Q and lucky losers LL - as defined in Section 2.3 - have zero h coefficients. Hence, they do not affect directly the O.F. Moreover, from a quantitative perspective, any two Q or LL players can not be distinguished because of their 0 impacts on O.F.

In the first instance, the solver is used as a *black box* to find θ different solutions with a time limit of t_l. The best two ones, $x^{\theta 1}$ and $x^{\theta 2}$ are compared. If a qualified player $i \in I$ is allocated to the same $j \in J$ cluster in both the two best solutions $x^{\theta 1}$ and $x^{\theta 2}$, the allocation becomes definitive. Hence, $x_{i,j}$ for that player is constrained to the value of 1. The maximum amount of qualified players fixed in each cluster is limited to Q_n/k, where Q_n is the number of qualified players in the whole problem. In addition to that, a *bucket* containing all the non-seeded and non-qualified players - which were stable in the best two solutions - is generated. Each cluster has its own bucket, and requirements to join it are reported in the following list.

1. A player must belong to the same cluster in the last two best solutions.
2. Given MD - the maximum degree of a player in the problem - only players with a *degree* ranging in $[\lambda_m MD, \lambda_M MD] : \lambda_m, \lambda_M \in [0,1]$ are added to the bucket.

The last requirement does not guarantee that the allocation is improving branched solutions. However, it reduces - to some extent - the degrees of freedom by fixing some players with conflict measures in a specified range. After generating the buckets, a randomly selected η percentage of players from the $bucket_j$ are fixed in the j^{th} cluster. The solver is used again with a time limit of $t_l \cdot 3/2$ and with $x^{\theta 1}$ as starting solution.

Since the variables are fixed once and only one sub-problem is created, this heuristic belongs to the class of *hard variable fixing* heuristics. This might lead to a sub-problem

with a bad approximation for the original optimum. There is no guarantee the new constrained solution space contains the original optimal value. Moreover, since players are fixed only if they are stable in the last two best feasible solutions, **a feasible solution is guaranteed** as well in $T \approx 5/2t_l$.

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;);
```

4.2.2 Heuristic B - weighted local branching

Local Branching was introduced by Fischetti & Lodi (2003). The solver is used as a *black bock* and the solution space is explored defining a neighborhood around a specific solution. While the method in Section 4.2.1 uses a *hard variable fixing* approach, local branching uses a *soft fixing* approach. Therefore, given a set of rules, the solver is able to decide which variable to fix and subsequently to analyze an easier sub-problem, without losing the chance of finding a better solution in the main problem.

The local branching is implemented in the tournament allocation problem with the reported notation. First, the solver is used to find a feasible solution $\bar{x}$, which is assumed as the incumbent one. The goal is to explore a neighborhood of $\bar{x}$. In particular a $K-OPT$ neighborhood which satisfies Equation 4.1. In order to avoid ambiguity, the k constant k for the local branching is renamed as k_{branch}.

$$\sum^{n}\sum_{x_{ij}=0}^{k} x_{ij} + \sum^{n}\sum_{x_{ij}=1}^{k} (1 - x_{ij}) \leq k_{branch} \tag{4.1}$$

The solver is consequently launched as to explore the left branch, represented by the new constraint in Equation 4.1. The right branch - which is complementary to the left - can be obtained with the constraint in Equation 4.2.

$$\sum^{n}\sum_{x_{ij}=0}^{k} x_{ij} + \sum^{n}\sum_{x_{ij}=1}^{k} (1 - x_{ij}) \geq k_{branch} + 1 \tag{4.2}$$

The original heuristic develops as follow. The solver explores the left branch with a time-limit t_l. $\Delta Z(x^{\theta}, x)$ - the first part of inequality 4.1 - is introduced as a measure of distance between the incumbent solution x^{θ} and a generic one x.

1. If **no improving solution** is found, the problem is led back to the right branch and the k_{branch} parameter is increased.
2. If an **improved solution** x^{θ} is found, it becomes the new incumbent solution $\bar{x} = x^{\theta}$, and the solver explores the right branch with the new incumbent as the initial solution.
3. If a **general time-limit** T_l is triggered, the last incumbent is the final solution
4. If an **optimal solution** is found for the general problem, the heuristic stops.

Starting from this local branching process, the proposed heuristic exploits players' degrees introduced in Section 4.1. The main assumption regards players with a *degree* ranging in $[\lambda_m \cdot MD, \lambda_M \cdot MD]$, in which MD is the maximum degree in the problem and $\lambda_m, \lambda_M \in$ [0,1]. The heuristic supposes those players have more inertia in moving between cluster than the other ones have. The increment in their inertia - compared to the one of regular players - is given by a factor of $\eta > 1$. Two subsets are created: players in $A \in I$ have more inertia, while $B \in I$ are the ones regular inertia, so that $A \cap B = \emptyset \wedge A \cup B = I$. Equation 4.1 is altered as follows:

$$\begin{aligned}\sum^{n}\sum_{x_{ij}=0 \wedge i \in B}^{k} x_{ij} + \sum^{n}\sum_{x_{ij}=1 \wedge i \in B}^{k} (1 - x_{ij}) \quad + \sum^{n}\sum_{x_{ij}=0 \wedge i \in A}^{k} \eta x_{ij} + \\ + \sum^{n}\sum_{x_{ij}=1 \wedge i \in A}^{k} \eta(1 - x_{ij}) \leq (\sum_{x_{ij}=1 \wedge i \in A}^{n} \eta) k_{branch}\end{aligned} \tag{4.3}$$

Intuitively, Equation 4.3 introduces a notion of weighted degree of freedom. A specific weight, which depends on the degree of the player, is assigned to each variable $x_{ij} \in A$. The search is limited to new $\widetilde{K} - OPT$ neighborhood which takes into account those weights. Therefore, modifying the value of a generic variable $x_{ij} \in A$ takes more effort than moving one $x_{ij} \in B$. In addition to that, the algorithm **switch to an unweighted local branching** in the case the solution is not improving or infeasible for more than 1 iteration.

Algorithm 4 presents a policy for managing the value of k_{branch}. As for the original local branching, $\Delta Z_w(\bar{x}, x)$ is a weighted distance between the incumbent solution and a generic one. The variable *switch* in Algorithm 4 makes possible to switch between the local branching and its weighted version. The k_{branch} parameter is incremented by $\mu > 1$ each time the solution is not improving or infeasible, in order to prevent stagnation in the same neighborhood. Moreover, if the solution is not improving after $ni - limit$ iterations, the heuristic stops and the last incumbent solution is adopted as best one. A set of arbitrary values for all the parameters is presented in Chapter 5.

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

Chapter 5

Implementation and simulations

5.1 Introduction

Experimental tests have been performed by using *Ilog CPLEX 12.7* running on a *3,5 GHz Intel Core i7* with *16 GB 2133 MHz LPDDR3* of RAM Memory, and operating system *MacOS 10.13.4*. The solver is integrated with *Java* and a *SQLite* database with the ATP data. Details about the dataset adopted are available in Section 8.3. The simulations are performed taking into account the four Grand Slams Tournaments from 2017: Roland Garros, Wimbledon, US Open and Australian Open. Further information about the source code is available in Chapter 8.
The following steps are taken in order to obtain results for a single tournament.

1. The tournament is loaded and the Matrix H is generated.
2. Two solutions are generated with $k = 4$.
   (a) A greedy solution built from the algorithm presented in Section 4.1.
   (b) One solution from CPLEX. This can be either an exact solution or one generated by a heuristic.
3. For each solution in 2, the following sub-process is iterated $s = u/2$ times.
   (a) Players are allocated in the first round with a draw.
   (b) A simulation for the full tournament moves on through the tournament until a tournament champion emerges.
   (c) Several indexes are tracked and measured both in the first round and next ones.

In order to give more solidity to the results, simulations are performed $u/2$ times. u is the number of unseeded players per cluster, as defined in Section 2.3.

5.2 Coefficients in Matrix H

Following the definition of Matrix H given in Section 2.3, we introduce a **conjectural set of rules** for conflicts and matrix H coefficients. Those rules can be defined as to fit specific needs for each tournament. The rules presented constitute only **one of the many reasonable options** and they are selected in order to fit the four Grand Slams. In the first instance, the default value of each element in H is $h_{\alpha,\beta} = 0 \ \ \forall \alpha, \beta \in I$. After matrix H is set to a zero-matrix, h coefficients are computed accordingly to the following rules.

Rule 1 If two players $\alpha, \beta \in I$ are from the same country, then $h_{\alpha,\beta} += 5;$.

Rule 2 If two players $\alpha, \beta \in I$ played against each other in a 1^{st} round in the last year, then $h_{\alpha,\beta} += 5;$.

Rule 3 If two players $\alpha, \beta \in I$ played against each other in a 2^{nd} round in the last year, then $h_{\alpha,\beta} += 2;$.

Rule 4 If two players $\alpha, \beta \in I$ played against each other in a 3^{rd} round in the last year, then $h_{\alpha,\beta} += 1;$.

Rule 5 If two players $\alpha, \beta \in I$ played against each other either in quarter-final or semi-final rounds in the last year, then $h_{\alpha,\beta} += 0.5;$.

In this case, the first rule is ensuring that **country diversity is increased** by stating that players from the same country generate a conflict. For instance, this can be overturned in the case the need is to pair players from the same country. This virtual rule might lead to a case in which at least one player per country goes to the second round.

5.3 Generating solutions

Two solutions are generated for each tournament. The first one is found by CPLEX, either using the only solver or adopting one heuristic. The second solution is built with the greedy approach explained in Section 4.1. Both solutions are taken into account as to generate draws and full tournament simulations.

5.4 First round draw

The tournament allocation problem results in a solution in which players are allocated in k clusters. Hence, there is still the need to perform a draw in order to plant players into the brackets graph, as mentioned in Section 1.3. Therefore, players are randomly allocated to a free slot s_i in the first round $r_i = 0$. The only constraint for the draw process regards the cluster of allocation for each player, which is given by the solution considered. Hence, the random allocation takes place in k different groups.

5.5 Simulation of full tournaments

Once a generic first round draw is performed, next rounds can be predicted as follow. A blend of current position in **ATP ranking** and the **Head2Head** from the last 2 years is taken into account as to estimate the odds of winning a match for a player. Given two generic players $\alpha, \beta \in I$ we define the probability of α of winning a match against β as:

$$\begin{aligned} P(\alpha,\beta) &= 0.65 \cdot P_{rank}(\alpha,\beta) + 0.35 \cdot P_{H2H}(\alpha,\beta); \\ P_{H2H}(\alpha,\beta) &= \frac{\#Wins\ of\ \alpha\ against\ \beta}{\#Matches\ of\ \alpha\ against\ \beta} \end{aligned} \tag{5.1}$$

The term $P_{rank}(\alpha,\beta)$ is equal to 1 if the ranking of α is greater than the ranking of β and 0 otherwise. Each time at least one between α, β is a qualified player, or more generally the two players have not yet matched, the term $P_{H2H}(\alpha,\beta)$ is 0. Then, the probability is computed as follow:

$$P(\alpha,\beta) = P_{rank} \tag{5.2}$$

Each full tournament simulation starts at the first round $r_i = 0$ and advances until a winner is found in the last round $t - 1 = r$. As to compute the $P_{H2H}(\alpha,\beta)$, only matches played in the last two years are taken into account: the range is dynamically selected depending on the tournament date.

5.6 Tracked indexes

Some indexes are tracked as to measure the extent of improvements between solutions. Since the data from the four Grand Slams is available also in terms of winners and losers for each round, the data is loaded as the actual tournament. Hence, other results are compared with actual measures in the considered tournament. The following table sums up the tracked indexes in simulations. Moreover, the reports in Chapter 6 are more synthetic and simplified.

In the case a heuristics is used with an execution time of T_l, the solution is compared with the one obtained with CPLEX - namely the *CPLEX comparator* - running with the same time limit T_l. This process is required in order to compare the results obtained from the heuristic with the one from the raw CPLEX. As for the greedy solution, the comparator is launched with a time limit T_l equal to the amount of time required by the greedy algorithm to generate the solution. The former procedure gives insights into the performance of the greedy compared to CPLEX. More detailed statistics are as well available in Chapter 6.

5.7 Parameters for Heuristics

Following the definitions in Sections 3 and 4, Tables 5.2 and 5.3 report the parameters set for heuristics used in simulations.

Table 5.1: Tracked indexes in simulations

#	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

Chapter 6

Computational testing

6.1 Matrix H

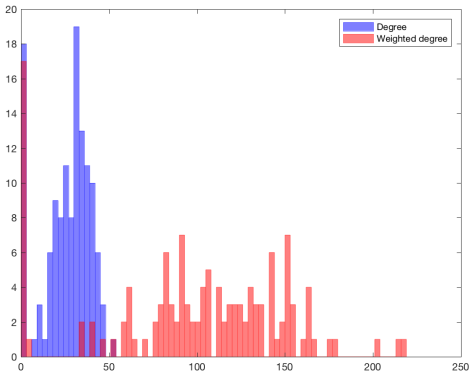
Some statistics and graphics from the 4 H-matrixes generated for the four Grand Slams of 2017 are reported in the following section. The rules for generation and attribution of h values are the ones reported in Section 5.2. Table 6.1 sums up several key parameters for matrixes H. Among those, the number of nodes having a degree in one of the ranges specified in heuristics - as of Algorithms in 3 and 4 - and the average weighted degree.

Table 6.1: Overview for matrixes H

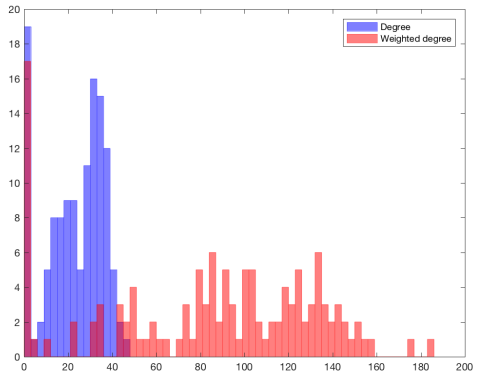
#	**RG**	**WI**	**US**	**AUS**
Average degree	25.66	22.53	22.47	24.14
Average weighted degree	97.30	84.01	87.16	88.93
Number of Q and LL	17	17	18	17
Max h in H	15.00	17.00	17.00	15.00
Max degree MD	54	46	47	48
Max weighted degree	217.50	183.50	191.00	201.00
Nodes with degree in [0.35MD,0.65MD]	66.00	39.00	48.00	53.00
Nodes with degree in [0.45MD,0.55MD]	14.00	14.00	18.00	14.00

6.1.1 Degree distributions

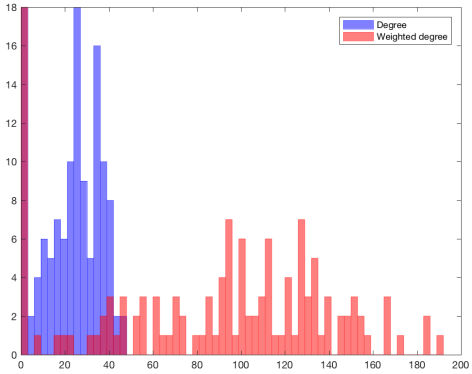
By looking at the Figures in 6.1, peaks on 0 are related to qualified players, which have no connections with other players. Moreover, the distribution of degrees and weighted degrees in tournaments seem to follow a normal distribution. Despite no deep investigations have been made, the Figures in 6.2 report the fit of those distributions with the normal one. By virtually excluding qualified players and the most connected ones, we obtain a good fit for the remaining players.



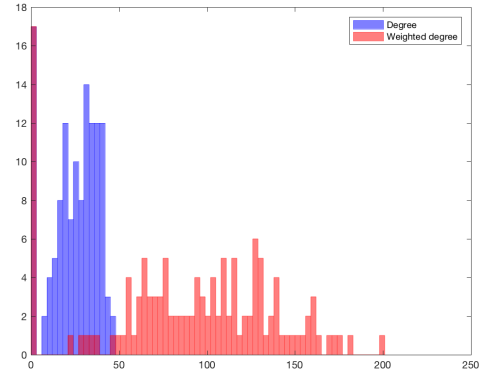
(a) Roland Garros



(b) Wimbledon

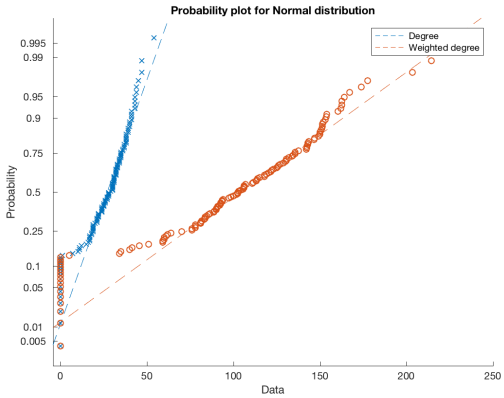


(c) US Open

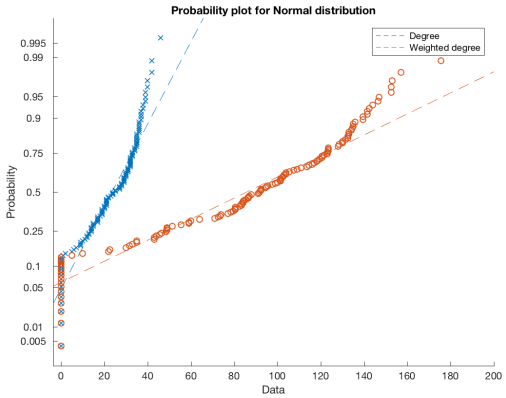


(d) Australian Open

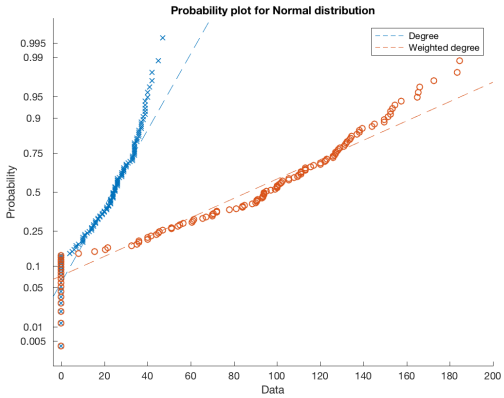
Figure 6.1: Distributions for degrees of players in Grand Slams of 2017



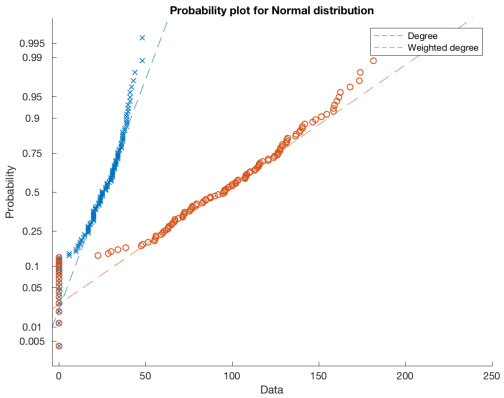
(a) Roland Garros



(b) Wimbledon



(c) US Open



(d) Australian Open

Figure 6.2: Fit to normal distribution for degree and weighted degree distributions in Grand Slams of 2017.

6.1.2 Graphs

The Figures in 6.3 represent matrix H with a circular graph. The complexity of the four tournaments is graphically evident because of the density of connections. Most of the players entertain several connections - or conflicts - with many others. Moreover, qualified players are easy to spot as they do not have conflicts. In order to report a graph similar to the one introduced with the max-cut problem in Section 3.1, Figure 6.4 represents a graph for the whole Wimbledon tournament. The unconnected nodes around the main agglomerate are qualified players, while the ones with names are the most connected ones. In particular, they are the one with a weighted degree in the top $67th$ percentile.

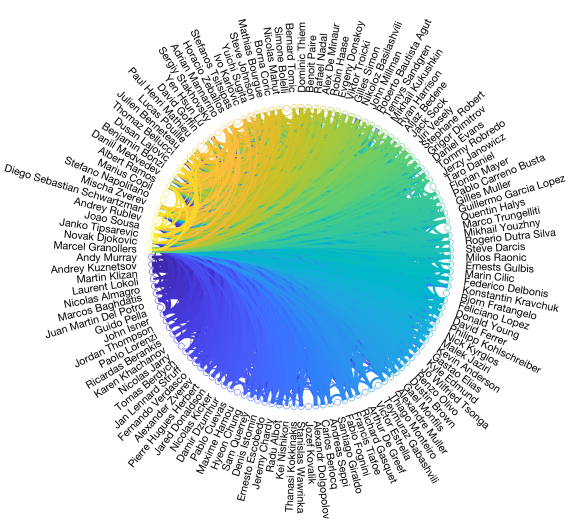
6.2 Grand Slam Tournaments results

The subsections above report results from simulations in which the CPLEX solution is the **optimal one**. Results reported in column ACT are for the actual tournament, CPLEX for the optimal solution from the solver, GREEDY from the one built up with the greedy algorithm. Values *in parenthesis* refer to the improvement - or worsening if negative - of the given index in comparison to the actual tournament.
Moreover, values for ACT across rows always refer to the real tournament. Therefore, even if they are reported as "average" they refer to the only real tournament.

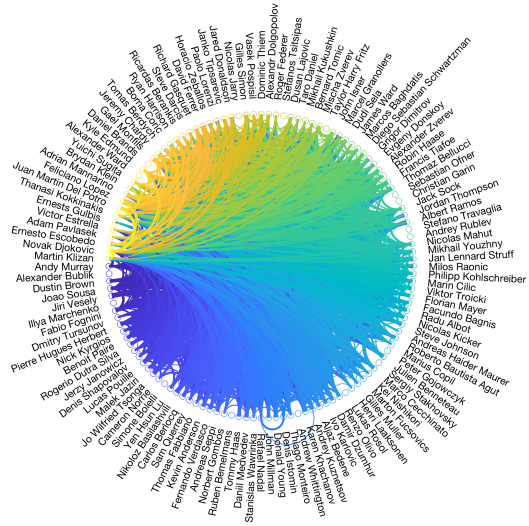
Table 6.2: Roland Garros results

	ACT	**CPLEX**	**GREEDY**
LB	142.0	142.0	142.0
Time	-	4,186.28s	1.88s
CPLEX O.F. with TL=GreedyTime	–	885.0	821.5
O.F.	1429.5	803.5 (43.79%)	821.5 (42.53%)
Greedy improvement with swaps	-	-	47.5
Solution status	-	Optimal	Feasible
Average indexes | s=16			
Number of conflicts in the 1st round	11	8	7.69
Measure of conflicts in the 1st round	41.5	25.72 (38.03%)	24.47 (41.04%)
Number of conflicts in the tournament	32	25.25	25.56
Measure of conflicts in the tournament	122.0	62.25 (48.98%)	63.38 (48.05%)
Last simulation indexes			
Number of conflicts in the 1st round	11	9	7
Measure of conflicts in the 1st round	41.5	25.5 (38.55%)	14.5 (65.06%)
Number of conflicts in the tournament	32	28	21
Measure of conflicts in the tournament	122.0	57 (53.28%)	35.5 (70.9%)

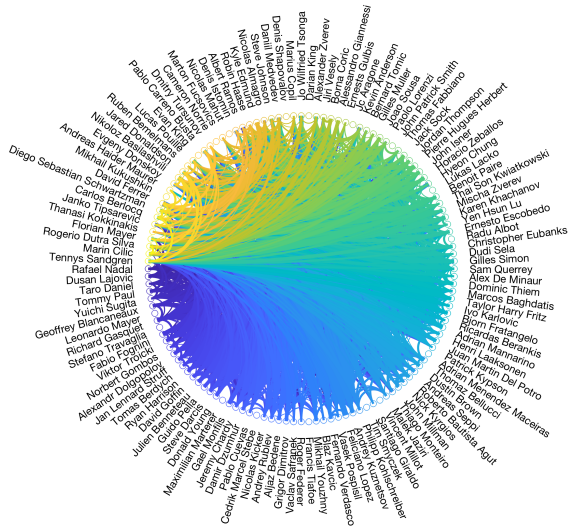
The results in the tables illustrate **remarkable reductions of the conflicts**. The most interesting result derives from the **greedy algorithm**, which is noticeably faster than any solver instance or heuristic. Moreover, there is some interesting evidence about



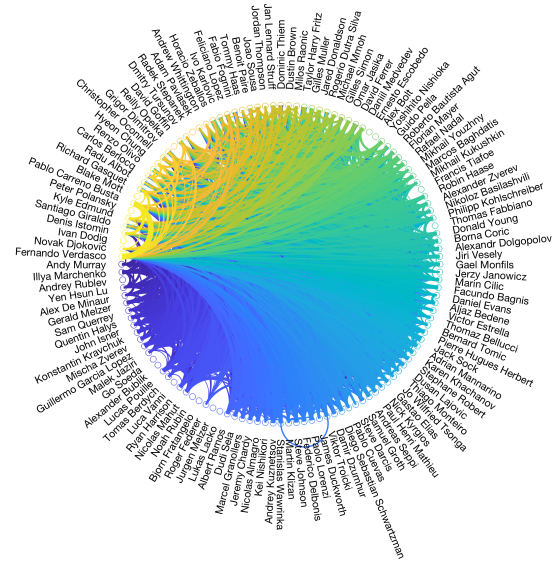
(a) Roland Garros



(b) Wimbledon



(c) US Open



(d) Australian Open

Figure 6.3: Circular graphs for Grand Slams of 2017

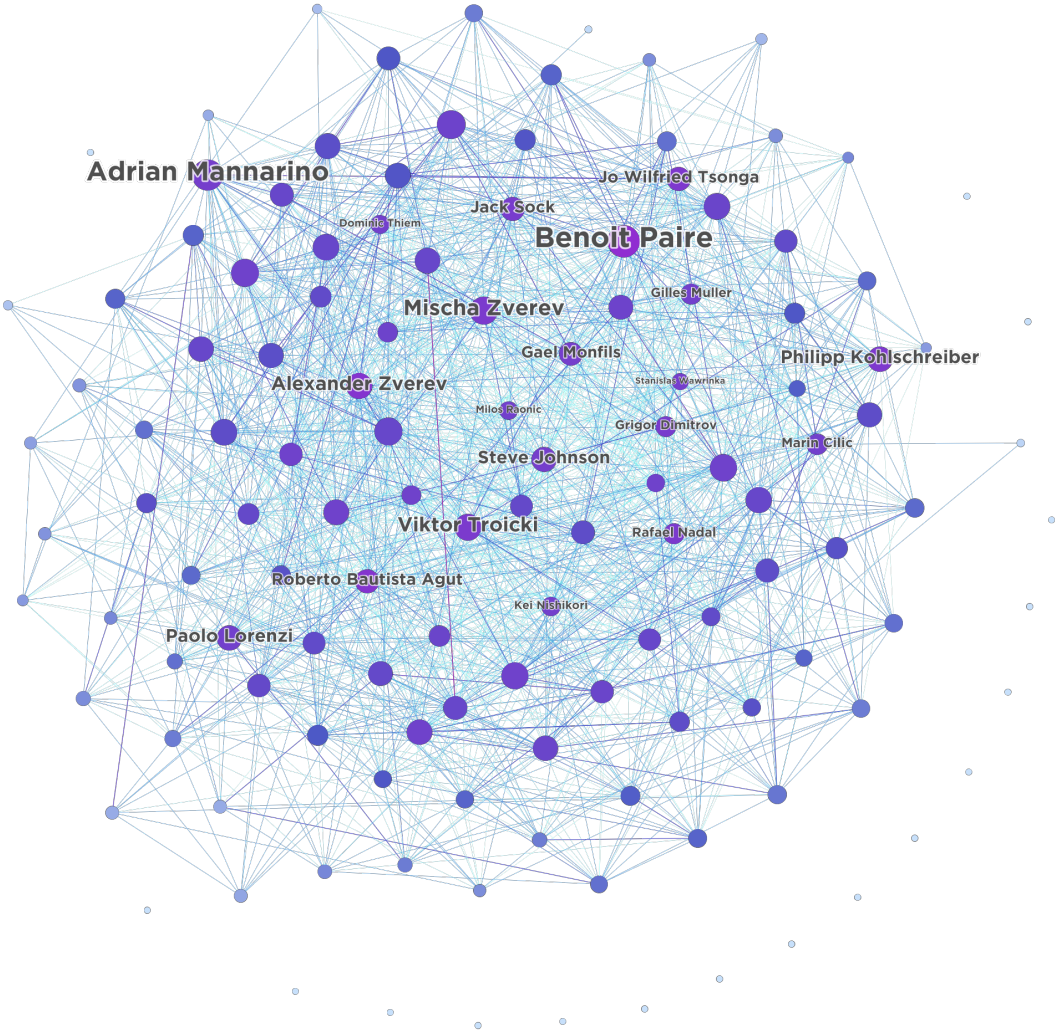


Figure 6.4: Wimbledon 2017 graph - Fruchterman Reingold g=5.0 computed with *Gephi*. Node size and node name is proportional to the weighted degree. Labeled nodes have a weighted degree in the $67th$ percentile.

Table 6.3: 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%)

Table 6.4: US Open results

	ACT	**CPLEX**	**GREEDY**
LB	164.5	164.5	164.5
Time	-	764.99s	1.88s
CPLEX O.F. with TL=GreedyTime	–	820.5	756.0
O.F.	1310.0	710.5 (45.76%)	756.0 (42.29%)
Greedy improvement with swaps	-	-	39.5
Solution status	-	Optimal	Feasible
Average indexes | s=16			
Number of conflicts in the 1st round	12	5.19	5
Measure of conflicts in the 1st round	48.5	16.94 (65.08%)	16.03 (66.95%)
Number of conflicts in the tournament	32	24.94	23.81
Measure of conflicts in the tournament	110.5	78.16 (29.27%)	76.53 (30.74%)
Last simulation indexes			
Number of conflicts in the 1st round	12	5	6
Measure of conflicts in the 1st round	48.5	13.5 (72.16%)	18 (62.89%)
Number of conflicts in the tournament	32	25	30
Measure of conflicts in the tournament	110.5	80 (27.6%)	96 (13.12%)

Table 6.5: Australian Open results

	ACT	**CPLEX**	**GREEDY**
LB	135.0	135.0	135.0
Time	-	3,933.37s	1.88s
CPLEX O.F. with TL=GreedyTime	–	828.5	751.5
O.F.	1328.5	715.0 (46.18%)	751.5 (43.43%)
Greedy improvement with swaps	-	-	70.5
Solution status	-	Optimal	Feasible
Average indexes | s=16			
Number of conflicts in the 1st round	12	7.44	7.94
Measure of conflicts in the 1st round	31.5	19 (39.68%)	22.06 (29.96%)
Number of conflicts in the tournament	29	27.44	28.06
Measure of conflicts in the tournament	76.0	64.53 (15.09%)	69.19 (8.96%)
Last simulation indexes			
Number of conflicts in the 1st round	12	8	9
Measure of conflicts in the 1st round	31.5	25 (20.63%)	20.5 (34.92%)
Number of conflicts in the tournament	29	29	31
Measure of conflicts in the tournament	76.0	65.5 (13.82%)	72 (5.26%)

the **overall improvement of conflicts** with this greedy approach. In fact, despite the O.F. does not reach the optimal value given by CPLEX, the conflicts improvement rate is very close to the one from the optimal solution.
Wimbledon and US Open tournaments require a relatively quick computation of the optimal solution. In such cases, the use of heuristic approaches might be less appealing. On the other hand, tournaments such as Australian Open and Roland Garros take more time to converge to the optimal solution.

6.3 Obtaining a brackets graph

Each simulation computed for the whole tournament ends with the name of the tournament champion. While simulating the four Grand Slams, winners and finalist were almost every time among the top and seeded players. This is basically due to the way the probability of winning is defined - as mentioned in Section 5.5. Moreover, this is also trivial because the top seeds are supposed to be the best players.
However, we can underline how the purpose of the tournament allocation approach is not acting on the tournament structure in a way that discourages the best players. Instead, the model is reaching its goals about diversity and fairness without strongly affecting finalists in last rounds, but rather by **generating new opportunities** for unseeded players. Hence, the names obtained in simulations are the ones that we normally expect to win those tournaments. The following output reports how a single Wimbledon tournament might evolve according to the model. Moreover, in Table 6.6 are reported the tournament

champions for the referenced simulations of Wimbledon 2017.

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

```
Round R16
        Fabio Fognini wins over Andy Murray
        Nick Kyrgios wins over Diego Sebastian Schwartzman
        Jo Wilfried Tsonga wins over Sam Querrey
        Stanislas Wawrinka wins over Fernando Verdasco
        Rafael Nadal wins over Karen Khachanov
        Gilles Muller wins over Marcel Granollers
        Roberto Bautista Agut wins over Kei Nishikori
        Steve Johnson wins over Robin Haase
        Albert Ramos wins over Aljaz Bedene
        Gilles Simon wins over Andreas Seppi
        Grigor Dimitrov wins over John Isner
        Roger Federer wins over Mischa Zverev
        Dominic Thiem wins over Paolo Lorenzi
        Tomas Berdych wins over Richard Gasquet
        Gael Monfils wins over Borna Coric
        Juan Martin Del Potro wins over Novak Djokovic
Round R8
        Nick Kyrgios wins over Fabio Fognini
        Stanislas Wawrinka wins over Jo Wilfried Tsonga
        Rafael Nadal wins over Gilles Muller
```

```
        Roberto Bautista Agut wins over Steve Johnson
        Albert Ramos wins over Gilles Simon
        Roger Federer wins over Grigor Dimitrov
        Dominic Thiem wins over Tomas Berdych
        Gael Monfils wins over Juan Martin Del Potro
Round R4
        Stanislas Wawrinka wins over Nick Kyrgios
        Rafael Nadal wins over Roberto Bautista Agut
        Roger Federer wins over Albert Ramos
        Dominic Thiem wins over Gael Monfils
Round R2
        Rafael Nadal wins over Stanislas Wawrinka
        Roger Federer wins over Dominic Thiem
Round R1
        Rafael Nadal wins over Roger Federer
The winner is: Rafael Nadal!
```

Listing 1: Last rounds for a single Wimbledon 2017 simulation.

6.4 Heuristics performances

All the four tournaments are tested with the two introduced heuristics. Five tests are performed for each heuristic and the average results are reported in Tables 6.7, 6.8, 6.9 and 6.10. As mentioned, heuristics aim to find qualitatively good solutions with a limited computational effort. In this respect, the results confirm that the CPLEX solver employs several good exploration policies and raw heuristics that are hard to outperform. However, the introduced heuristics can be useful for less performing solvers which do not embed effective exploration policies. At the same time, we remark that the results for the two heuristics are not as solid as the ones obtained with the greedy algorithm. Hence, further improvements and changes might be required.

Table 6.7: Heuristics on Roland Garros

TOURNAMENT NAME	**Roland Garros**
PROBLEM OPTIMAL O.F.	803.50
PROBLEM OPTIMAL EXECUTION TIME	4186.28s
INSTANCES PER HEURISTIC	5
HEURISTIC A	
Execution time	14.07s
O.F. Value	875.50
Solution status	Feasible
Pure CPLEX O.F. Value	868.50
HEURISTIC B	
Execution time	72.26s
O.F. Value	834.00
Solution status	Feasible
Pure CPLEX O.F. Value	824.50

Table 6.8: Heuristics on Wimbledon

TOURNAMENT NAME	**Wimbledon**
PROBLEM OPTIMAL O.F.	500.5
PROBLEM OPTIMAL EXECUTION TIME	660.0s
INSTANCES PER HEURISTIC	5
HEURISTIC A	
Execution time	8.89s
O.F. Value	721.50
Solution status	Feasible
Pure CPLEX O.F. Value	712.00
HEURISTIC B	
Execution time	42.19s
O.F. Value	688.00
Solution status	Feasible
Pure CPLEX O.F. Value	686.00

Table 6.9: Heuristics on Us Open

TOURNAMENT NAME	**UsOpen**
PROBLEM OPTIMAL O.F.	710.50
PROBLEM OPTIMAL EXECUTION TIME	764.9s
INSTANCES PER HEURISTIC	5
HEURISTIC A	
Execution time	9.62s
O.F. Value	752.50
Solution status	Feasible
Pure CPLEX O.F. Value	729.25
HEURISTIC B	
Execution time	91.35s
O.F. Value	716.50
Solution status	Feasible
Pure CPLEX O.F. Value	714.50

Table 6.10: Heuristics on Australian Open

TOURNAMENT NAME	**Aus Open**
PROBLEM OPTIMAL O.F.	715.00
PROBLEM OPTIMAL EXECUTION TIME	3933.37s
INSTANCES PER HEURISTIC	5
HEURISTIC A	
Execution time	9.89s
O.F. Value	767.50
Solution status	Feasible
Pure CPLEX O.F. Value	775.50
HEURISTIC B	
Execution time	73.27s
O.F. Value	745.50
Solution status	Feasible
Pure CPLEX O.F. Value	728.50

Chapter 7

Final remarks

The evidence about match repetitions has been the fundamental starting point of this work. Only after addressing the existence of potential conflicts between players, a formalization for the tournament allocation problem (TAP) has been given in a formal way.
The TAP seeks to allocate players in disjoint clusters with the aim of avoiding conflicts and increasing diversity across matches in a specific single elimination tournament. Therefore, players have the opportunity to fairly dispute matches with new rivals and are positioned in the brackets graph with a **more fair approach**. The peculiar characteristics for any specific tournament can be reflected in the way fairness is defined through the model. Furthermore, we might speculate the **public can enjoy a more diverse and varied set of matches** in tournaments created with this approach. One of the most important elements is the stochastic component of the model, both mathematically and in terms of sports ethics. It ensures that tournaments are subject to the *luck of the draw*.

The combinatorial optimization approach has led to concrete and measurable results, which improved the overall conflict measures in the tested tournaments. Moreover, the two introduced heuristics provide a quicker way to obtain good solutions with resource-constrained machines, less efficient solvers and a limited amount of time.
The most interesting result comes from the **greedy algorithm**, which achieved remarkable results in short times. Looking beyond the combinatorial approach, some interesting observations about degrees and networks linked to the analyzed tournaments have been made. Some techniques of **data visualization** have been applied in order to make those findings more appreciable. In particular, we underline that the process which led to the development of the greedy approach rooted in the visualization of degree distributions. Without any optimization tool, the greedy approach builds a qualitatively good solution - as demonstrated by results in Chapter 6 - in no more than few seconds.

Tennis tournaments are complex networks of links between players and may turn out to be difficult to approach with pure solvers, in terms of time and computational resources. Hence, the greedy algorithm plays a fundamental role in providing a viable way to ensure quick and good results. As to visualize the complexity laying behind

those networks, some graphs have been included in the previous section. Moreover, an interactive web visualizer for the four Grand Slams is available in Section 8.2.

Despite this allocation model might challenge the almost pure stochastic method in use nowadays, results show that **diversity and fairness are improving**. As mentioned in the introduction, dealing with fairness in a broad meaning can be complex. Thus, the key is defining the rules and principles behind the concept. The results also show how the new allocation process is *not affecting finalists* in the last rounds but rather can **create new opportunities** in early rounds for unseeded players. The increased diversity among matches is positive for players as well as for the public. And winners - as it should be - are always among the best players. Evidently, the simulation part does not comprehend phenomena such as new emerging players, hence can not predict outcomes in which new talented players are joining the tournament. On the other side, new possibilities are created for those players as to climb the ladder.
As the final outline, the tournament allocation problem can be framed into several applications from different fields. Because its deep links with the max-cut problem and the max-diversity, the same approach can be applied in many different disciplines.

Chapter 8

Appendix

8.1 Source code

The full source code is available on GitHub on:

`https://github.com/gdragotto/TournamentAllocationProblem`

The source includes the following parts:

1. *Main.java* contains the main source code. This includes the model implementation, procedures to load data from the database, heuristics, a solution greedy approach and a full-tournament simulators.

2. *MatrixH.java* contains the code as to generate the Matrix H following the rules given in Section 6.1. In addition to that, the function to computer the odds of winning matches - as referenced in Section 5.5 - is implemented in the class.

3. *Player.java* contains the structure for the object *Player*.

4. *Results* folder contains all the instances with optimal solutions.

5. *Matlab* folder contains the MATLAB® code wrote to analyze matrixes H and compute some graphs.

6. *Gephi* folder contains the Gephi files as to render full tournament graphs.

8.2 Interactive visualizer

The four Grand Slams tournaments can be visualized at the following web page:

`https://gdragotto.github.io/TournamentAllocationProblem/`

8.3 Data from ATP

In order to run simulations, the ATP database from Sackmann (2017) of TennisAbstract.com is used. Four ATP's Grand Slams Tournaments from 2017 are taken in analysis: Roland Garros, Wimbledon, Australia Open and US Open. The choice is due to the vast amount of data available for both players and tournaments. Seeding positions and qualified players are integrated into the dataset with official data from the ATP's website on ATP (2018).

Bibliography

ATP (2018), 'Draws | atp world tour'.
URL: *www.atpworldtour.com*

Della Croce, F. (n.d.), 'Mixed integer linear programming models for combinatorial optimization problems', *Concepts of Combinatorial Optimization, Wiley-ISTE, 2nd Edition* pp. 101–133.

Della Croce, F., Grosso, A. & Salassa, F. (2013), 'Matheuristics: embedding milp solvers into heuristic algorithms for combinatorial optimization problems', *Heuristics: Theory and Applications* pp. 31–52.

Fischetti, M. & Lodi, A. (2003), 'Local branching', *Mathematical programming* **98**(1-3), 23–47.

Horen, J. & Riezman, R. (1985), 'Comparing draws for single elimination tournaments', *Operations Research* **33**(2), 249–262.

Kartik, S. & Karimi, S. (2007), 'Max-cut problem'. NC State University.
URL: *http://www4.ncsu.edu/ kksivara/ma796s/projects/sahar_report.pdf*

Kuo, C.-C., Glover, F. & Dhir, K. S. (1993), 'Analyzing and modeling the maximum diversity problem by zero-one programming', *Decision Sciences* **24**(6), 1171–1185.

Sackmann, J. (2017), 'Tennis atp-database'.
URL: *https://github.com/JeffSackmann/tennis_atp*

Sackmann, J. (2018), 'Trivia: Deja vu all over again'.
URL: *www.tennisabstract.com/blog/2018/03/01/trivia-deja-vu-all-over-again/*

Williams, V. V. (2010), Fixing a tournament., Vol. 1, Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), AAAI, pp. 895–900.

Wimbledon (2017), 'Seeds - the wimbledon championship'.
URL: *https://www.wimbledon.com/en_GB/atoz/seeds.html*