Matrik: Jurnal Manajemen, Teknik Informatika, dan Rekayasa Komputer
Vol. 23, No. 1, November 2023, pp. 113∼130
ISSN: 2476-9843, accredited by Kemenristekdikti, Decree No: 200/M/KPT/2020
DOI: 10.30812/matrik.v23i1.3290

Single Elimination Tournament Design Using Dynamic Programming Algorithm

Yusri Ikhwani[1], Asary Ramadhan[1], Muhammad Bahit[2], Taufik Hidayat Faesal[3]
[1] Universitas Islam kalimantan Muhammad Arsyad Al Banjari, Banjarmasin, Indonesia
[2]Politeknik Negeri Banjarmasin, Banjarmasin, Indonesia
[3]Universiti Tun Hussein Onn, Johor, Malaysia

Article Info

Article history:

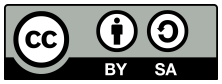
Received August 21, 2023
Revised September 10, 2023
Accepted October 15, 2023

Keywords:

Algorithm
Dynamic Programming
Optimization
Single Elimination
Tournament

ABSTRACT

Finding the best single-elimination tournament design is important in scientific inquiry because it can have major financial implications for event organizers and participants. This research aims to create an optimal single-elimination tournament design using binary tree modeling with dummy techniques. Dynamic programming algorithms have been used to compute optimal single-elimination designs to overcome this effectively. This research method uses various implementations of sub-optimal algorithms and then compares their performance in terms of runtime and optimality as a solution to measure the comparison of sub-algorithms. This research shows that the difference in relative costs produced by various sub-algorithms with the same input is quite low. This is expected because quotes are generated as integer values from a small interval 1, ≤ 9, whereas costs tend to reach much higher values. From the comparison of these sub-algorithms, the best results among the sub-optimal algorithms were obtained in the Sub Optimal algorithm 3. We present the experimental findings achieved using the Python implementation of the suggested algorithm, with a focus on the best single-elimination tournament design solution.



Corresponding Author:

Yusri Ikhwani, +6282152745971,
Information Technology and Informatic,
Universitas Islam Kalimantan Muhammad Arsyad Al Banjari, Banjarmasin, Indonesia.
Email: as.ary29@gmail.com

How to Cite:
yusri ikhwani, M. Bahit, A. Ramadhan, and T. Faesal, "Single Elimination Tournament Design Using Dynamic Programming Algorithm", *MATRIK: Jurnal Managemen,Teknik Informatika, dan Rekayasa Komputer*, Vol. 23, No. 1, pp. 113-130, Nov, 2023.

1. INTRODUCTION

A knockout tournament, also called a single-elimination tournament, is a format employed in sporting events to ascertain the victor of a match [1, 2]. In a knockout tournament, contestants compete against each other according to an initial seeding and advance to subsequent rounds by defeating their immediate adversaries. The outcome of each match determines a solitary victor from the two players who have contested. Draws are not considered in this context. The exploration of the ideal tournament structure holds a key position in scholarly inquiry due to its far-reaching implications. Unoptimal configurations can profoundly impact the financial aspects of event organizers, participating individuals, and teams and even influence the preferences of the match spectators, thereby garnering substantial attention from researchers in this field [3]. To illustrate, teams engaged in the UEFA Champions League receive substantial monetary incentives as they progress through different tournament phases. Therefore, making the correct selection of players (which must be done before the commencement of the knockout rounds) assumes paramount importance. The economic significance of sports continues to expand, as indicated by a European Commission report citing 2012 figures, which states that the sport-related contribution to the Gross Domestic Product (GDP) of the European Union stood at 2.12%, equivalent to €279.7 billion [4]. While tournaments are often associated with sports, politics is another significant domain where tournaments play a crucial role, particularly in organizing elections. However, for this paper's themes, the spotlight will primarily be on sports tournaments. Fundamentally, tournament design revolves around two primary principles: the "round robin" approach and the knockout principle. These principles can be employed independently or in conjunction to create diverse tournament structures, contingent on factors like the player count, available time, and the specific context in which the tournament is being held. The knockout tournament is organized hierarchically, resembling a binary tree structure, where the leaf nodes symbolize individual players or teams. In contrast, internal nodes signify the matches within the tournament [5]. This competition unfolds through a sequence of rounds, commonly known as match rounds [6]. When the total count of players, denoted as P, is a power of 2, such as $P = 8 = 2^3$, the tournament structure forms a complete binary tree, with all participants engaged in the initial round.

The primary driving factor behind this research stems from the fact that previous investigations have predominantly centered on the optimal design of knockout tournaments when the number of participants is a power of 2 [7–9]. However, in practical scenarios, the number of players might not adhere to this power of 2 constraint, as exemplified by the case when $P = 9$. Certain players will be granted exemptions in such instances, allowing them to directly participate in the second round of matches without undergoing the initial round. Bădică [10] formally conceptualizes competition through a single-elimination tournament, commonly known as a knockout tournament. This design aims for optimality by enhancing its appeal, ensuring that top-ranking players have the chance to face each other at the advanced stages of the tournament. They also introduced a Dynamic Programming algorithm for the computation of optimal tournaments. They analyzed its algorithmic complexity, leading to a comprehensive examination of solution optimality and the efficiency of its runtime performance. Optimization of the tournament design using utility functions optimized with simulated annealing algorithms and optimal Bayesian design carried out by Hennessy [11]. Guyon [12] demonstrated the effective utilization of this tournament format to maximize the number of knockout matches within the context of UEFA Champions League matches. The design was coined "Choose Your Opponent" by the author. Theoretical considerations also arise when examining potential outcomes within knockout tournaments. This study explores upper and lower bounds on player winning probabilities within random knockout tournaments [13]. The focus here pertains to examining random knockout tournaments, wherein a random process governs the determination of match outcomes in each round. Moreover, this research assumes that the probabilities of victory and defeat for every match involving two players are established. Conducted a theoretical exploration of knockout tournaments [14] with a specific emphasis on evaluating scenarios where the number of players is a power of 2. However, sports like MuayThai [15], tennis, chess, and soccer often involve a number of players that cannot be categorized as powers of 2. Nonetheless, this study will not delve into the analysis of particular sports; its sole focus lies on the optimal design of knockout tournaments. Tournaments find numerous practical applications, particularly within fields like sports. An intriguing discourse on sports economics from an operational research standpoint and practical applications is elucidated in a recent scholarly publication by Csató [16]. This discourse centers on several tournament ranking paradoxes and provides concrete illustrations of the implementation of tournament design.

Based on the related work, there has been no discussion [2, 13, 14] about optimal calculations using the dummy technique in the structure of single-system tournaments. As a point of differentiation from similar research, we introduce a structured definition of competitive events here, taking the shape of a single-elimination tournament, commonly known as a knockout tournament. We extend this concept using the dummy technique and present a Dynamic Programming Algorithm to efficiently compute optimal tournaments in various scenarios as a contribution to this study. Additionally, we analyze the algorithm's complexity in detail. The objective of this research is to create an optimal single elimination tournament design using binary tree modeling with dummy technique. To address this work dynamic programming algorithm has used for effectively calculating optimal single elimination design.

This paper is organized as follows: Section 1 introduces the single elimination tournament and concept design using the dummy technique problems. Section 2 describes the research method used to describe the contributed solution methodology. Results

and analysis are presented in Section 3. Section 4 includes the conclusion and future work. Section 5 acknowledgment. Section 6 declarations are made in the last section.

2. RESEARCH METHOD

In this study, we will harness mathematical models and techniques guided by the notion of proof. Our approach involves representing the tournament through a binary tree structure, where each leaf node represents an individual player, and each internal node represents a match between two players and determines the victor. In constructing the binary tree brackets, we will systematically derive this proposition from a foundational set of principles, referred to as definitions. Definition 1 is stated as follows.

2.1. Binary Tree Brackets

Definition 1. Let us contemplate a finite nonempty set of players, denoted as σ. We establish the set of trees, denoted as T(σ), with serving as the set of leaf nodes, based on the subsequent criteria (1) If consists of a single element, denoted as i, the set T(σ) consists of a single tree with a single node representing i. (2) if σ_1 and σ_2 are two separate sets of players, we define the set σ as the union of σ_1 and σ_2 ($\sigma = \sigma_1 \bigcup \sigma_2$). We represent it as the following Equation (1).

$$T(\sigma) = \{tree \mid tree = \{tree_1, tree_2\}, tree_1 \in \sigma_1, tree_2 \in \sigma_2\} \tag{1}$$

It's worth noting that the set notation in Equation (1) indicates that the trees are unordered, meaning that the arrangement of the left and right branches is not significant. We examine instances of tournaments involving sets of players containing 1, 2, 3, and 4 elements. This can be observed in the illustrative example provided in Example 1.

1. If the set σ contains only the element 1, then $T(\sigma) = \{1\}$.
2. If σ is defined as the set containing the elements 1 and 2, then $T(\sigma) = \{\{1, 2\}\}$.
3. If σ is defined as the set containing the elements 1, 2, and 3, then applying the function T to σ would result in a set that contains nested sets. Specifically, it would be represented as $T(\sigma) = \{\{\{1, 2\}, 3\}, \{\{1, 3\}, 2\}, \{\{3, 2\}, 1\}\}$.
4. If is defined as the set containing the elements 1, 2, 3, and 4, then applying the function T to σ would result in a collection of nested sets, forming different tree structures ($T(\sigma) = \{\{\{1, 2\}, \{3, 4\}\}, \{\{1, 3\}, \{2.4\}\}, \{\{1, 4\}, \{2, 3\}\}, \{\{\{1, 2\}, 3\}, 4\}, \ldots\}$). In this particular case, it can be observed that there are 15 distinct trees.

Graphical representations of certain tournaments introduced in Example 1 are depicted in Figure 1. Notice that the tournaments in the top row (labeled "a" and "b") consist of a number of elements that is a power of two ($2 = 2^1$) and $4 = 2^2$, respectively and are fully balanced. Nevertheless, the tournaments in the second row are not entirely evenly matched. Even though the lower rightmost tournament includes $4 = 2^2$ players, it doesn't qualify as fully balanced. From an intuitive perspective, the tournament with three players (labeled "c") seems reasonable because player 3 joins the tournament just one round after players 1 and 2, implying a sense of "balancing." However, the lower rightmost tournament with four players (labeled "d") is deemed unacceptable since player 4 is exempted from playing in the first two rounds, which is considered unfair.

Next, we will employ the statement "counting binary tree bracket," or it can be referred to as a proposition. Proposition 1 for counting binary brackets is expressed in Equation (2). The set $T(\sigma)$, where $\mid \sigma \mid$ is P players, contains Equation (2) elements.

$$\frac{(2P-2)!}{(P-1)!.2^{P-1}} \tag{2}$$

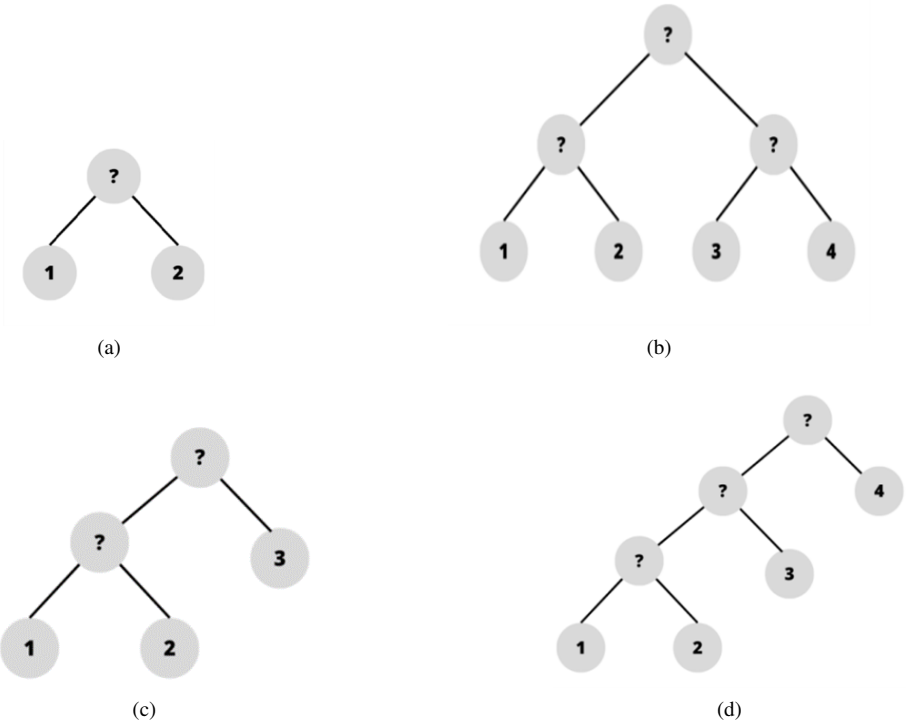


Figure 1. Binary tree brackets of two and three players (first column) and four players (second column), (a) Balanced binary tree brackets with two players, (b) Balanced binary tree brackets with four players, (c) Balanced binary tree brackets with three players, (d) Balanced binary tree brackets with four players

The number of full binary tree brackets with P leaves is equal to $C_{(P-1)}$, where C_P represents Catalan number [17]. The method for determining the proof of Proposition 1, as represented in Equation (2), is expressed in Equation (3).

$$C_p = \frac{1}{(P+1)} \begin{pmatrix} 2P \\ P \end{pmatrix} \tag{3}$$

Now, every permutation of the P players can be linked to the leaf nodes of a binary tree, resulting in $P! \cdot C_{(P-1)}$ trees. Nonetheless, the branches of every internal node can be swapped, leading to an identical tree. There are $P-1$ internal nodes, which means there's a total of $2(P-1)$ swaps, resulting in the number of trees given by Equation (4):

$$\frac{P! \cdot C_{p-1}}{2^{P-1}} = \frac{(2P-2)!}{(P-1)! \cdot 2^{P-1}} \tag{4}$$

In Example 2, we will demonstrate that our calculations are consistent with Example 1 discussed earlier. For example, If P is equal to 3 ($P=3$), we obtain a total of $\frac{4!}{2!*2^2} = 3$ trees, and when P is equal to 4 ($P=4$), we obtain $\frac{6!}{3!*2^3} = 15$ trees. A legitimate tournament needs to be equitable, meaning that each player should engage in (approximately) an equal number of games to secure victory.

Upon examining the competition presented in Example 1 and Figure 1, it becomes clear that when the size of the set $|\sigma| \leq 3$ is considered. Every element in $T(\sigma)$ corresponds to a legitimate tournament. However, when the set has a size of 4 ($|\sigma| = 4$), it can be observed that only 3 trees within $T(\sigma)$ truly depict valid competitions. For instance, take the example of $\{\{1,2\},\{3,4\}\}$, which represents a valid competition. In this scenario, each player must participate in exactly two games to have a chance of winning the competition. In this instance, we have a completely equitable tournament comprising $P = 2^2$ players. Furthermore, $\{\{1,2\},3\}$ is also considered a valid competition, meeting the criteria where players 1 and 2 must engage in two games to win, while player 3 needs just one game for victory. This means that player 3 is not required to participate in the first round, leading to a maximum disparity of one game played among the players. Nevertheless, $\{\{\{1,2\},3\},4\}$ fails to satisfy the conditions for a legitimate competition. In

this scenario, players 1 and 2 must participate in three games to secure victory, while player 4 only requires one game to win. This leads to a disparity of more than one game played by each player, surpassing the fairness threshold. In this context, having multiple exemptions for a player is deemed unfair.

It's noticeable that in a legitimate competition, the tree's configuration exhibits a feature where all its leaf nodes are at a height of either R or $R+1$, with R being a specific value. The determination of R can be deduced from the given player count, P, in the competition, and it signifies the total number of rounds encompassed by the competition.

Let's examine a tournament with R rounds. It's easy to recognize that the highest number of players, $P_{max} = 2^R$, and this occurs when the first round consists of a maximum of 2^{R-1} games. Therefore, we have Equation (5) for a tournament with R rounds. Note that Equation (5) implies by given Equation (6).

$$2^{R-1} < P \leq 2^R \tag{5}$$

$$R = [log_2 P] \tag{6}$$

Once again, we are creating a valid and balanced binary tree definition. We refer to this definition as Definition 2. Consider the variable R, where R belongs to the set of natural numbers ($\mathbb{N}$) and signifies the number of rounds in the competition. Let σ denote a nonempty set comprising P players that meet conditions Equation (5) and (6). The set $T_R(\sigma)$, representing balanced trees with R layers, is defined as the assortment of balanced (valid) competitions with R rounds in the following manner:

1. When R is equal to 0 ($R = 0$), it implies that P equals 1 ($P = 1$). Consequently, we have a singleton set $\sigma = \{i\}$. In this particular scenario, $T_0(\sigma)$ is equivalent to the set $\{i\}$ ($T_0(\sigma) = \{i\}$).
2. If R is greater than or equal to 1, and there exist $t_1 \in T_{R-1}(\sigma_1)$ and $t_2 \in T_{R-1}(\sigma_2)$ such that $\sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup \sigma_2 = \sigma$, then $= \{t_1, t_2\} \epsilon T_R(\sigma)$.
3. If R is greater than or equal to 2, and there exist $t_1 \in T_{R-1}(\sigma_1)$ and $t_2 \in T_{R-2}(\sigma_2)$ represents a fully balanced tree with $\mid \sigma_2 \mid = 2^{R-2}$, and $\sigma_1 \cap \sigma_2 = \emptyset$ with their union being equal to σ, then $t = \{t_1, t_2\} \epsilon T_R(\sigma)$

If R is greater than or equal to 1, then the number of players $P \in 2^{R-1} + 1 \ldots 2^R$. A tree in $t \in T_R(\sigma)$ can be created by either (i) combining two balanced trees with $R-1$ layers or (ii) combining one balanced tree with $R-1$ layers and one fully balanced tree with $R-2$ layers (where all the leaves are located on layer $R-2$). In both scenarios, the balancing condition of t is maintained appropriately.

We refer to the proposition concerning the brackets of a balanced competition as Proposition 2. Consider a competition $t \in T_R(\sigma)$, where σ represents a set of P players, and the value of R is determined according to Equation (6). The initial round of the competition begins with $C = 2P - 2^R$ players, while the number of dummy players starting the first round is $D = 2^R - P$ to transform a perfect binary tree into a full binary tree. Furthermore, when R is greater than or equal to 1, the number of level 2 internal nodes in the tree can be expressed as $E = P - 2^{R-1}$, which is equal to the number of players P. This implies that $C = 2E$, and $D = 2^{R-1} - E$. The proof of this proposition is as follows:

First, let us note that if the number of players is a power of 2, indicated by $P = 2^R$, then we have $E = 2^{R-1} = P/2, C = P$, and $D = 0$. This statement is straightforwardly true, as in such cases, the competition is perfectly balanced, and all players commence the competition in the first round without any exemptions or byes. The general case can be proven using an induction technique based on $R \in \mathbb{N}$. When R equals 0, the competition consists of P = 1 player. In this scenario, a solitary balanced competition exists where $D = 0$ and $C = 1$, satisfying the property straightforwardly. For $R = 1$, the competition has $P = 2$ players. A single balanced competition exists in this case, characterized by $E = 1$, $C = 2$, and $D = 0$. Therefore, the property is straightforwardly fulfilled. Now, let's assume that the property holds for $x = 0, 1, \ldots, R$. We will proceed to prove that it also holds for $x = R + 1$.

1. Case 1. If $R \geq 1, t_1 \in T_R(\sigma_1), t_2 \in T_R(\sigma_2), \sigma_1 \cap \sigma_2 = \emptyset$ and $\sigma_1 \cup_2 = \sigma$, consider $t = \{t_1, t_2\} \in T_{R+1}(\sigma)$, where σ is a nonempty set, and t fulfills the second condition stated in Definition 2. According to the induction hypothesis, we can assume that $D_i = 2^R - P_i, C_i = 2P_i - 2^R, E_i = P_i - 2^{R-1}$ for $i = 1, 2$ and $P = P_1 + P_2$. Then $D = D_1 + D_2 = 2^{R+1}(P_1 + P_2) = 2^{R+1}P$. Similarly, we can show that $C = C_1 + C_2 = 2P2^{R+1}$ and $E = E_1 + E_2 = P2^R$. This verifies the desired result.
2. Case 2. If $R \geq 2, t_1 \in T_R(\sigma_1), t_2 \in T_{R-1}(\sigma_2)$ is a fully balanced tree (i.e., $|\sigma_2| = 2^{R-1}$) $\sigma_1 \cap \sigma_2 = \emptyset, and \sigma_1 \cup \sigma_2 = \sigma$. We will now examine $t = \{t_1, t_2\} \in T_{R+1}(\sigma)$ such that the third condition of Definition 2 is satisfied. According to the induction hypothesis, we have $D_1 = 2^R - P_1, C_1 = 2P_1 - 2^R, E_1 = P_1 - 2^{R-1}, D_2 = 0, C_2 = 2^{R-1}$ and $E_2 = 2^{R-2}, P_2 = 2(R1)$, and $P = P_1 + 2^{R-1}$. So $C = C_1 = 2(P_1 + 2^{R-1}) - 2^R - 2^R = 2P - 2^{R+1}$ and also $D = D_1 + C_2 = 2^R - P_1 + 2^{R-1} = 2^R + 2^{R-1} + 2^{R-1}P = 2^(R+1)P$ and also $E = E_1 = P_1 - 2^{R-1} = P - 2^{R-1} - 2^{R-1} = P - 2^R$. Thus, it has been demonstrated.The relationships $C = 2E$ and $D = 2^{R-1} - E$ can now be easily verified.

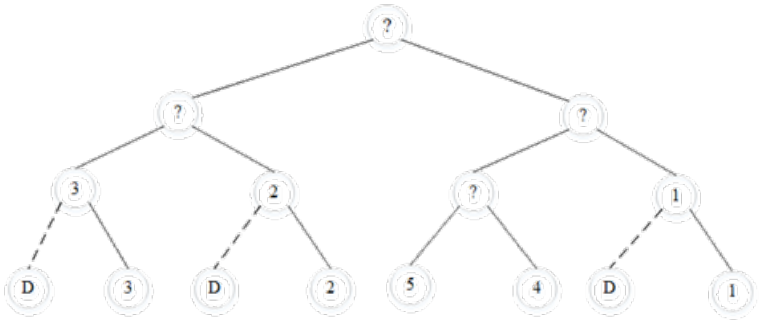


Figure 2. Balanced hierarchically shaped single-elimination brackets with 3 dummies and five real players

We describe the binary tree design for the match brackets in Example 3. Consider a competition with $P = 5$ players. In this scenario, we have $R = 3, D = 2^3 - 5 = 3, E = 2^2 - 3 = 1, C = 2 * 5 - 2^3 = 2$. The corresponding tree structure will have three layers in a competition involving five players. The first layer will consist of $C = 2$ leaves (representing players), and the second layer will consist of 2 internal nodes and $D = 3$ leaves (representing players) among the total of $2^{R-1} = 4$ nodes in that layer. One example of a balanced competition is illustrated in Figure 2, where D means that represents a dummy player.

We propose to create a proposition for counting balanced hierarchically shaped single elimination brackets. We refer to this proposition as Proposition 3. The set $T_R(\sigma)$ with $|\sigma| = P$ players contains $\frac{P!.\begin{pmatrix} 2^{R-1} \\ D \end{pmatrix}}{2^{P-1}}$ elements. For a proof, there are $\begin{pmatrix} 2^{R-1} \\ D \end{pmatrix}$ ways to choose how the D players will progress to the second round. Their arrangement is significant, so we multiply by $D!$. Furthermore, those D players are selected arbitrarily from the set of P players, so we multiply by $\binom{P}{D}$. In the end, the arrangement of those C remaining players who enter the first round is significant, so we also multiply by $C!$. For every inner node in the tree, swapping its left and right sub-tree maintains a tournament invariant. There are 2^{P-1} distinct methods to switch the left and right sub-trees of the tree, so we need to divide by 2^{P-1}.

2.2. Single elimination best design

The matchups between players are determined in each round of a competition consisting of R rounds. It is crucial to emphasize that in a competition, two players can only face each other once in a match during a particular round. This property arises because in a binary tree, any two leaves (nodes without children) have a distinct common ancestor, the closest shared ancestor between them. As a consequence, the competition round $X_{i,j}$ where players i, j can face each other is a distinct and well-defined value within the range of $1, 2, \ldots R$. For instance, taking into account the competition depicted in Figure 2, $X_{3,5} = 3, X_{1,4} = 2, X_{4,5} = 1$. In an intuitive sense, it is preferable for players i and j with higher rankings to face each other in the later stages of the competition. This arrangement increases the significance and intensity of their matches.

In the following discussions, we assume that there is a quotation $r_i \in (0, +\infty)$ available for each of the i $\epsilon\sigma$ players. In the context mentioned, quotations can be obtained based on the players' current rankings, similar to how it is done in international Boxing and Muaythai competitions. Alternatively, quotations can be acquired through other methods or criteria. Definition 3 (Brackets cost). Consider $t \in T_R(\sigma)$, which represents a competition with R rounds. Let $X_{ij}^t \in \{1, 2, \ldots, R\}$, indicating the stage in which players i, j can potentially meet in the competition t. Assume $r_i > 0$ represents the quotatioxxns of players for all $i \in \sigma$. The cost of t is defined as follows:

$$Cost(t) = \Sigma_{i,j \in \sigma, i<j} r_i r_j X_{ij}^t \tag{7}$$

Definition 4 (Optimal brackets). A single-elimination competition bracket refers to a competition where its cost, as computed using Equation (7), is maximized. The definition of a single-elimination competition brackets is determined by:

$$\begin{aligned} OptimalBracket(\sigma) &= max_{t \in T_R(\sigma)} Cost(t) \\ t^* &= argamax_{t \epsilon T_R(\sigma)} Cost(t) \end{aligned} \tag{8}$$

It is evident that players with higher rankings are associated with higher quotations. We assume that the quota r_i of a player, i corresponds to their rank s_i, such that if $s_i < s_j$, it follows that $r_i > r_j$. For instance, in the case where there are $P = 2^R$ players, we can assign $r_i = P + 1 - s_i$ as the quota for each player $i = 1, \ldots, P$. Example 4. We will consider three competitions,t_1, t_2, and t_3, with three players each. The players are shown in Figure 3. We will now introduce some new terms:

$$\begin{array}{rcl} L & = & r_1r_2 + r_1r_3 + r_2r_3 \\ M & = & r_1 + r_2 + r_2 \end{array} \tag{9}$$

We obtain:

$$\begin{array}{rcl} Cost(t_1) & = & r_2r_31 + r_1r_22 + r_1r_32 = L + r_1(r_2 + r_3) = L + r_1(M - r_1) \\ Cost(t_2) & = & L + r_2(M - r_2) \\ Cost(t_3) & = & L + r(M - r_3) \end{array} \tag{10}$$

The arrangement of the costs is determined by the sequence of values of the function $r(M - r)$ for $r = r_1, r_2, r_3 \in [0, M]$. The function exhibits a monotonic increase on $[0, M/2]$ and a monotonic decrease on $[M/2, M]$. If the condition $r_i \leq M/2$ holds, which means that no player receives more than half of the total quotation stake, then the arrangement of the costs is determined by the order of the quotations r_i.

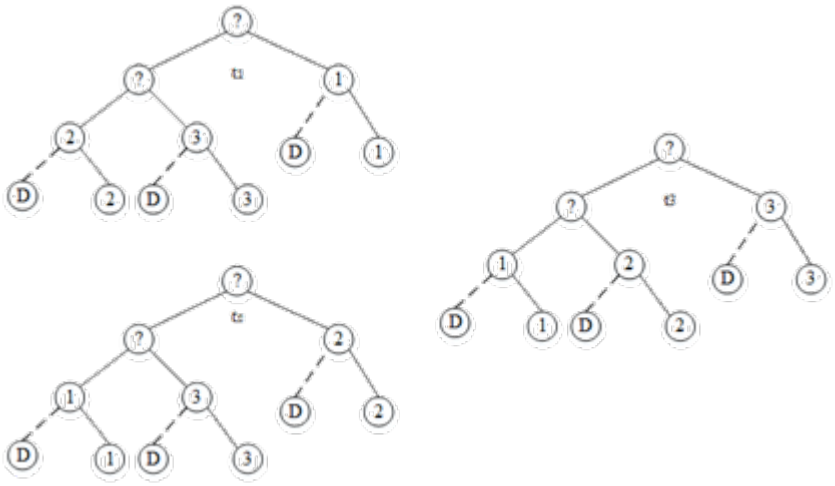


Figure 3. Balanced hierarchically shaped single-elimination competition brackets with 3 dummy players and three real players

Example 5. Let us examine a situation involving four players, as presented in Table 1. Each player is assigned a distinct rank from 1 to 4. If we use the $r_i = 5 - s_i$ approach to determine players' quotas, player 2, who has rank 4, will be assigned a quota $r_2 = 1$. We consider the three competitions $t_1, t_2, t_3 \in T_2(\{1, 2, 3, 4\})$ from Figure 4. The cost of a competition $t \in T_2(\{1, 2, 3, 4\})$is calculated according to Equation (7), as follows:

$$\begin{array}{rcl} Cost(t) & = & \Sigma^4_{1 \leq i < j \leq 4} r_i r_j X^t_{ij} \\ Cost(t) & = & 4 \times 1.X^t_{12} + 4 \times 2.X^t_{13} + 4 \times 3.X^t_{14} + 1 \times 2.X^t_{23} + 1 \times 3.X^t_{24} + 2 \times 3.X^t_{34} \end{array} \tag{11}$$

By substituting the stage values X^t_{ij} for each competition in Table 2 in Equation (11), we can obtain the cost values of the competitions in Table 2. It can be observed that, in this case, the best competition is t_3. After conducting a thorough analysis, it can be confirmed that competition t_3 is the optimal choice when the quota values are arranged in decreasing order according to ranks.

Table 1. Ranking and quotation of players for $R = 4$

Player i	**Rank** (S_i)	**Quota** $r_i = R + 1 - S_i$
1	1	4
4	2	3
3	3	2
2	4	1

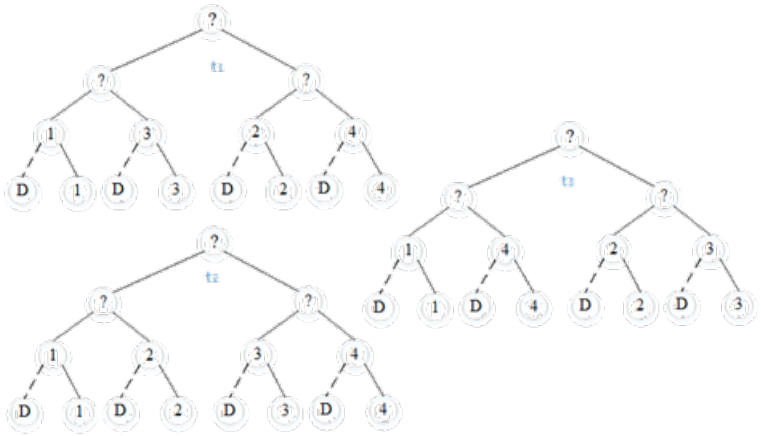


Figure 4. Balanced hierarchically shaped single-elimination competition brackets with 2 dummy players and four real players

Table 2. Stages of gameplay and corresponding costs for each competition in Figure 4

X^{t_1}	**1**	**2**	**3**	**4**	X^{t_2}	**1**	**2**	**3**	**4**	X^{t_2}	**1**	**2**	**3**	**4**
1		2	2	1	1		2	2	1	1		1	2	2
2			1	2	2			1	2	2			2	2
3				2	3				2	3				1
Cost			56				56				60			
			Cost of t_1				Cost of t_2				Cost of t_3			

2.3. Best single elimination design with dynamic programming

The dynamic programming algorithm can be implemented in either a bottom-up approach or a top-down approach with memoization [18]. The comprehensive research was given by Bădică et al. in [19], who provided valuable insights and proposed a dynamic programming algorithm that could be implemented using both a top-down approach with memoization and a bottom-up approach. This algorithm calculates the single-elimination competition brackets and their associated cost.

Proposition 4 (Recurrences for optimal brackets competitions cost). The optimal brackets competition cost $OptimalBracket(\sigma)$ introduced by Equation (8) can be defined recursively as follows:

$$OptimalBraket(\sigma) = \begin{cases} 0 & R = 0, |\sigma| = 1 \\ r_i.r_j & R = 1, \sigma = \{i, j\} \\ max\ OptimalBracket(\sigma_1) + OptimalBracket(\sigma_2) + Rr\sigma_1 r\sigma_2 & R \geq 2, 2^{R-1} < |\sigma| \leq 2^R \\ \sigma_1 \cup \sigma_2 = \sigma & 2^{R-2} \leq |\sigma_1| \leq \\ \sigma_1 \cap \sigma_2 = \emptyset & |\sigma_2| \leq 2^{R-1} \end{cases} \quad (12)$$

The optimal brackets competition can be determined by monitoring the pairs of subsets $Optimal(\sigma) = (\sigma_1, \sigma_2 = \sigma \setminus \sigma_1)$ that maximize $OptimalBracket$ in Equation (12) for a given value of $R \geq 2, 2^{R-1} < |\sigma| \leq 2^R$ as described below:

$$OptimalSub(\sigma) = argmax OptimalBracket(\sigma_1) + OptimalBracket(\sigma_2) + Rr\sigma_1 r\sigma_2 \sigma_1, \sigma_2 \subseteq \sigma$$

Moreover, the sets $OptimalSub(\sigma)$ can be used to construct a single-elimination competition bracket. Let $\sigma^R = \sigma$ represent the set of all possible subsets of σ^R. We define a function $(\sigma^{R-1} = OptimalSub(\sigma^R)), \ldots, \sigma^0 = OptimalSub(\sigma^1)$ to represent the optimal cost of a competition for a given subset σ^R. Based on this, the optimal bracket competition t^* can be recursively defined as follows:

$$
\begin{aligned}
t^* &= t^R(\sigma^R) \\
t^i(\sigma^i) &= \begin{cases} \{t_1^{i-1}(\sigma^{i-1}), t_2^{i-1}(\sigma^i \setminus \sigma^{i-1})\} & i \geq 1 \\ j & i = 0, \sigma^0 = j \end{cases}
\end{aligned}
$$

The following sections will investigate these options by deriving a bottom-up dynamic programming algorithm specifically designed for fully balanced competitions. Additionally, we will explore a top-down dynamic programming algorithm with memoization that can handle the general case.

2.4. Top-down Dynamic Programming Algorithm with Memoization

Proposition 6 determined by assuming that we want to calculate $OptC(\Sigma)$ for $|\Sigma| = P, P \geq 2, 2^{n-1}+1 \leq P \leq 2^n$. According to proposition 5, we must recursively explore all tournaments of shape $t = \{t_1, t_2\}$ such that $t_i \in T(\Sigma_i), |\Sigma_i| = P_i, i = 1, 2, P = P_1 + P_2$.

$$
\begin{gathered}
max\{P - 2^{n-1}, 2n - 2\} \leq P_1 \leq \lfloor P/2 \rfloor \\
P = P_1 + P_2 \text{ and } P_1 \leq P_2
\end{gathered} \tag{13}
$$

Obtained Equation (14) :

$$
\begin{gathered}
P_1 \leq \lfloor P/2 \rfloor \\
\text{As } P = P_1 + P_2 \text{ and } 2^{n-2} P_1 \leq 2^{n-1}
\end{gathered} \tag{14}
$$

Obtained Equation (15):

$$
\begin{gathered}
2^{n-2} \leq P_1 \leq 2^{n-1} \\
P - 2^{n-1} \leq P_1 \leq P - 2^{n-2}
\end{gathered} \tag{15}
$$

Combination between Equations (14) and (15), obtained Equation (16) :

$$
max\{P - 2n - 1, 2n - 2\} \leq P_1 \leq min\{2^{n-1}, P - 2^{n-2}, \lfloor P/2 \rfloor\} \tag{16}
$$

By merging the conclusions drawn from Propositions 5 and 6, we arrive at the top-down methodology for calculating optimal balanced tournaments, as outlined in Algorithm 1.

Algorithm 1: OptTourCostTD (Σ, P, q) **top-down dynamic programming algorithm with memoization for computing the cost of the optimal tournament.**

Global:	OptC, initially $\emptyset$, maps subsets of players to costs of optimal tournaments.
	OptS, initially $\emptyset$, maps subsets to sub-subsets for building optimal tournaments.
Input:	$P \in N^*$represents the number of players.
	q. Vector of size N representing the players quota
	Σ such that $\|\Sigma\| = P$. Σ represents the set of players.
Output:	C_{max}, Cost of the optimal tournament for set Σ of players.

$n \longleftarrow [log_2 P]$
if $n = 0$ **then**
    $C_{max} \longleftarrow 0$
    $S_{max} \longleftarrow \emptyset$
else if $n = 1$(i.e., $\Sigma = \{i, j\}$) **then**
    $C_{max} \longleftarrow q_i^* q_j$
    $S_{max} \longleftarrow \{i, j\}$
else
    $PL1 \longleftarrow 2^{n-2}$
    $PL2 \longleftarrow 2^* PL1$
    $k_{max} \longleftarrow \lfloor P/2 \rfloor$
    if $P \leq PL1 + PL2$ **then**
        $k_{min} \longleftarrow PL1$
    else
        $k_{min} \longleftarrow P - PL2$
    end if
    $C_{max} \longleftarrow \infty$
    for k_{min}, k_{max} **do**
        for $\Sigma_1 \subseteq \Sigma s.t \mid \Sigma_1 \mid = k$ **do**
            if $\Sigma_1 \notin OptC$ **then**
                $C_1 \longleftarrow OptTourCostTD(\Sigma_1, k, q)$
            else
                $C_1 \longleftarrow OptC \mid \Sigma_1 \mid$
            end if
            $\Sigma_2 \longleftarrow \Sigma \setminus \Sigma_1$
            if $\Sigma_2 \notin OptC$ **then**
                $C_2 \longleftarrow OptTourCostTDC(\Sigma_2, k, q)$
            else
                $C_2 \longleftarrow OptC \mid \Sigma_2 \mid$
            end if
            $C \longleftarrow C_1 + C_2$
            $ql \longleftarrow 0$
            for $i \epsilon \Sigma_1$ **do**
                $ql \longleftarrow ql + q_i$
            end for
            $qr \longleftarrow 0$
            for $i \epsilon \Sigma_2$ **do**
                $qr \longleftarrow qr + q_i$
            end for
            $C \longleftarrow C + k^* ql^* qr$
            if $C > C_{max}$ **then**
                $C_{max} \longleftarrow C$
                $S_{max} \longleftarrow \Sigma_1$
            end if
        end for
    end for
end if
$OptC \mid \Sigma \mid \longleftarrow C_{max}$
$OptS \mid \Sigma \mid \longleftarrow S_{max}$

2.5. Sub-Optimal Algorithm

In a broader sense, the dynamic programming algorithm strategy for formulating optimal tournaments adheres to the top-down divide-and-conquer approach elucidated in Algorithm 2.

Algorithm 2: OptTourCostSubOpt (Σ, P, q, S) **top-down divide-and-conquer algorithm for computing a sub-optimal tournament.**

Input: $P \in N^*$ represents the number of players.
q. Vector of size N representing the players quota
σ such that $\mid \Sigma \mid = P.\Sigma$ represents the set of players.
$S = \Sigma_{i \epsilon \Sigma} q_i$. Sum of players quotations
Output: C_{max}, Cost of the sub-optimal tournament for
set Σ of players.
t_{max}. sub-optimal tournament tree

$n \longleftarrow [log_2 P]$
if $n = 0$(i.e., $\sigma = \{i\}$) **then**
$C_{max} \longleftarrow 0$
$t_{max} \longleftarrow i$
else if $n = 1$(i.e., $\Sigma = \{i, j\}$) **then**
$C_{max} \longleftarrow 0$
$t_{max} \longleftarrow \{i, j\}$
else
$PL1 \longleftarrow 2^{n-2}$
$PL2 \longleftarrow 2^* PL1$
$k_{max} \longleftarrow \lfloor P/2 \rfloor$
if $P \leq PL1 + PL2$ **then**
$k_{min} \longleftarrow PL1$
else
$k_{min} \longleftarrow P - PL2$
end if
$C_{max} \longleftarrow -\infty$
for $\Sigma_1 \in Strategy(\Sigma, k_{min.k_{max}})$ **do**
$S_1 \longleftarrow 0$
for $i \in \Sigma_1$ **do**
$S_1 \longleftarrow S_1 + q_i$
end for
$k \longleftarrow \mid \Sigma_1 \mid$
$(C_1, t_1) \longleftarrow OptTourCostSubOpt(\Sigma_1, k, q, S_1)$
$(C_2, t_2) \longleftarrow OptTourCostSubOpt(\Sigma \setminus \Sigma_1, P - k, q, S - S_1)$
$C \longleftarrow C_1 + C_2 + k^* S_{1^*}(S - S_1)$
if $C > C_{max}$ **then**
$C_{max} \longleftarrow C$
$t_{max} \longleftarrow (t_1, t_2)$

end if
end for
end if
return

Note that when s equals 1, the time complexity of Algorithm 2 remains linear concerning the player count P. Furthermore, when s exceeds 1, the algorithm's time complexity becomes polynomial in P, and the polynomial's degree escalates logarithmically in relation to s.

3. RESULT AND ANALYSIS

3.1. Implementation

In essence, the subsets representing combinations of l elements from a set with R (where $R \geq l$) elements consist of all the permutations with repetitions of a binary vector consisting of R elements, where exactly l elements are equal to 1. Our experimental implementation of Algorithms 1, 2, and 3 required us to address several issues.

First and foremost, we chose to represent sets using arrays of bits and utilized the integer value equivalent to the binary representation as an array of bits. Secondly, to generate subsets of a given size (i.e., combinations), we utilized Algorithm 7.2.1.2L from [20], which generates permutations with repetitions of binary arrays. Thirdly, we had to select an efficient representation for the OptimalBracket and OptimalSub structures. The proper handling of these structures is essential for the effective implementation of some of our algorithms. Regarding our implementation, we have chosen the Python platform and opted to use subset-indexed dictionaries to represent OptimalBracket and OptimalSub. These dictionaries serve the purpose of mapping subsets of to their respective costs and the subsets required for constructing single-elimination competition bracketss, respectively. The dictionary keys representing the subsets are defined as integer values derived from their characteristic vectors in binary format. Since Python dictionaries are efficiently implemented using hash tables, the lookup operations are expected to have an average time complexity of O(1).

Lastly, for implementing the random selection of subsets, we used the array of bits representation of sets and employed the *random.permute* function from the NumPy package to generate a randomly permuted array representing a random subset.

3.2. Experiment Results

The research used a computer platform with the specification based on Intel Core i3 2.30 GHz CPU, 8 GB RAM, and Microsoft Windows 10 Profesional 64 Byte. The software uses Python 3.7.3 with a web-based interactive computing platform, namely Jupyter Notebook. Based on our research, no existing algorithms directly correspond to our proposed approaches. This can be attributed to two primary reasons: 1) We adopt an integrated approach to competition design instead of employing separate stages for structure design and seeding, which sets our approach apart from existing algorithms. 2) Our cost function lacks probabilistic information, making it difficult to directly compare competition cost values with algorithms that rely on such data. We pursued a distinct approach to assess our propositions. We developed optimal algorithms and various sub-optimal algorithm versions and compared their results regarding execution time and optimality. Consequently, we implemented and experimentally evaluated a total of eight algorithms, which are detailed in Table 3.

Table 3. Table showing the roster of implemented optimal and sub-optimal algorithms for balanced competition brackets.

Algorithm	Description	Optimality	Number of Players
OptimalDPTD	dynamic programming algorithm with memoization for computing the cost of the optimal brackets (Algorithm 1)	Optimal	Any natural number of choice
OptimalDPBU	dynamic programming-bottom-up for computing the cost of the optimal fully balanced brackets (Algorithm 2)	Optimal	Precise power of 2.
OptimalSubD1	Deterministic top-down divide-and-conquer strategy and unchanged quotation sequence that computes the sub-single-elimination competition brackets brackets (Algorithm 4)	Sub-optimal	Any natural number of choice
OptimalSubD2	Deterministic top-down divide-and-conquer strategy and increasingly sorted quotation sequence that compute the sub-single-elimination competition brackets brackets (Algorithm 4)	Sub-optimal	Any natural number of choice
OptimalSubD3	Deterministic top-down divide-and-conquer strategy and decreasingly sorted quotation sequence that computes the sub-single-elimination competition brackets (Algorithm 4)	Sub-optimal	Any natural number of choice
OptimalSubS1	The stochastic sub-optimal approach, Algorithm 4, using the stochastic strategy with $P_{sample} = 1$.	Sub-optimal	Any natural number of choice
OptimalSubS2	The stochastic sub-optimal approach, Algorithm 4, using the stochastic strategy with $P_{sample} = 1$.	Sub-optimal	Any natural number of choice
OptimalSubS3	The stochastic sub-optimal approach, Algorithm 4, using the stochastic strategy with $P_{sample} = 1$.	Sub-optimal	Any natural number of choice

It is important to highlight that the optimal algorithms face at least two constraints that make a comprehensive experimental comparison challenging with the other algorithms. First and foremost, their significantly high computational complexity restricts

their practical applicability to scenarios involving only a few players. Additionally, the dynamic programming bottom-up algorithm is restricted to scenarios where the number of players is an exact power of 2. We have only verified its performance for P = 4, 8, and 16.

In our dataset, we have multiple sequences of players' quotations. For each P value ranging from 3 to 50, we generated a sequence of quotations $(v_1, v_2, \ldots, v_P)$ with integer values v_i randomly selected from a uniform distribution within the interval $(v_{min} = 1, v_{max} = 9)$. We employed this dataset to conduct experimental evaluations of the algorithms listed in Table 3. All sequences in the dataset were utilized to test algorithms $(OptimalSubD_i)$ and $(OptimalSubS_i)$ for $i = 1, 2, 3$. We evaluated the optimal algorithm OptimalDPTD only for sequences corresponding to $P = 3, \ldots, 16$ players due to its high computational complexity. To control the running time of each problem instance, we set a limit of 8 minutes. We assessed the optimal algorithm $OptimalDPBU$ solely for sequences corresponding to P = 4, 8, and 16 players. This limitation is attributed to the algorithm's high computational complexity and the requirement that it works only with a number of players with an exact power of 2. For each algorithm, we documented both the (sub-)optimal cost of the resulting competition and the corresponding execution time. The stochastic algorithms ($OptimalSubS_i$ as $i = 1, 2, 3$) were evaluated by running them 13 times for each input sequence of quotations from the dataset. We recorded the minimum, maximum, and average costs and the average execution time.

Figure 5 displays the sub-optimal costs generated by the $OptimalSubD_i$ and $OptimalSubS_i$ algorithms for $i = 1, 2, 3$. The figure illustrates the costs Co_i obtained from $OptimalSubD_i$ algorithms for $i = 1, 2, 3$, along with the average costs $COSa_i$ generated by $OptimalSubS_i$ algorithms for $i = 1, 2, 3$, and the maximum cost $CoMAX_3$ derived from 14 repetitions of the $OptimalSubS3$ algorithm. We included only the maximum cost in this scenario since the stochastic algorithm ($OptimalSubS_i$ for $i = 1, 2, 3$) is expected to produce the best results, considering it uses the highest number of samplings ($P_{sample} = 3$). Upon examining Figure 5, we can observe that the relative cost difference generated by the different algorithms on the same input sequence is relatively low. As anticipated, this occurs because the quotations v_i were randomly generated as integer values within a small interval ($1 \leq v_i \leq 9$), while the cost can result in significantly higher values.

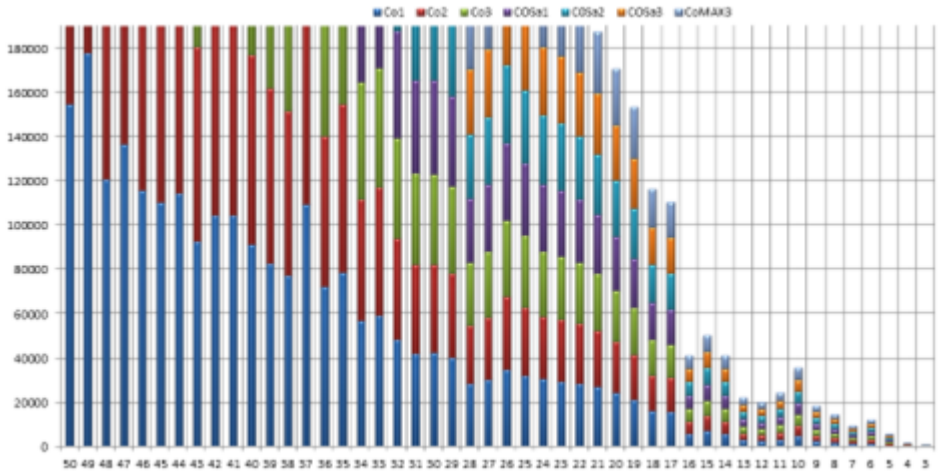


Figure 5. The sub-single-elimination competition brackets bracket costs obtained from different quotations' sequences using sub-optimal algorithms for different numbers of players.

For instance, when analyzing the results obtained for the sample with P = 38 players, we can observe that the relative difference between the lowest and highest costs achieved (70,321 and 79,451) is only 5.23%. It is worth noting that the sub-optimal algorithm $OptimalSubS3$ achieved the best results, while the algorithms $OptimalSubD2$ and $OptimalSubD3$ produced the worst results among the sub-optimal algorithms. It might appear somewhat unexpected that algorithm $OptimalSubD1$ outperforms $OptimalSubD2$ and $OptimalSubD3$; however, this can be attributed to the way the data set was generated. Algorithm OptimalSubD1 employs a random permutation of the quotations' sequence, leading to a more balanced distribution of total quotations between the subsets and $\sigma \setminus \sigma_1(seeAlgorithm4)$ compared to algorithms $OptimalSubD2$ and $OptimalSubD3$. One final observation, also confirmed experimentally, is that algorithms $OptimalSubD2$ and $OptimalSubD3$ produce identical sub-optimal costs when the number of players is an exact power of 2 (e.g., 16 and 32 in Figure 5). This outcome is a direct consequence of the logic behind their strategy definition.

Figure 6 illustrates the execution times TD_i and TS_i of algorithms $OptimalSubD_i$ and $OptimalSubS_i$ for $i = 1, 2, 3$. The time values are presented in milliseconds on a logarithmic scale, and they were obtained by calculating the average of 12 executions of each algorithm on the same input data. Initially, it is noticeable that the deterministic versions ($OptimalSubD_i$) are the fastest, and they exhibit nearly identical running times. This can be readily explained by implementing their underlying strategies, which have a low computational complexity. In essence, their strategies employ the same mechanism, with the additional sorting of the quotation sequence incurring a negligible cost, as it is executed before the core processing of the algorithms. Furthermore, as anticipated, $OptimalSubS3$ exhibits the highest execution time among the algorithms. This algorithm demonstrates the highest computational complexity among the sub-optimal algorithms due to its utilization of three subset samples during the top-down search. Based on Figure 6, it can be observed that the sequence of quotations with $P = 37$ players achieved the longest average execution time, reaching a value of 3.11 seconds. Figure 7 displays the outcomes generated by the optimal algorithms $OptimalDPTD$ and $OptimalDPBU$, along with a comparison to the results obtained from the sub-optimal algorithm $OptimalSubS3$, covering $P = 3, 4, \ldots, 16$ players in our input dataset.

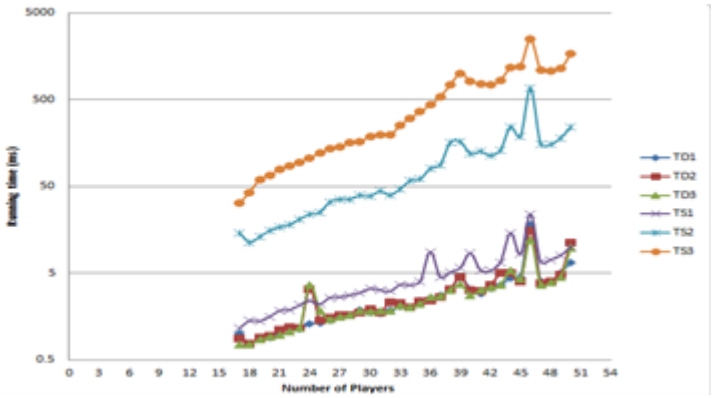


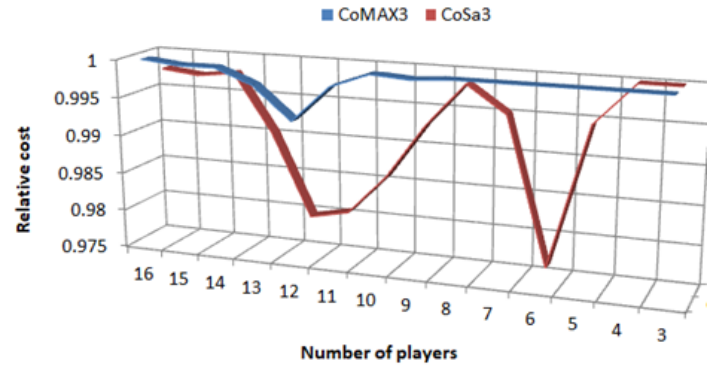
Figure 6. Graphs showing the logarithmic running times scale for sub-optimal algorithms on various quotations' sequences with varying numbers of players.

Figure 7a illustrates the contrast between the relative maximum and average costs achieved by algorithm $OptimalSubS3$ ($CoMAX3$ and $COSa_3$) and the true optimal cost attained by algorithm OptimalDPTD. The relative sub-optimal cost is represented by $D_s \in [0, 1]$ and is calculated using Equation (17).

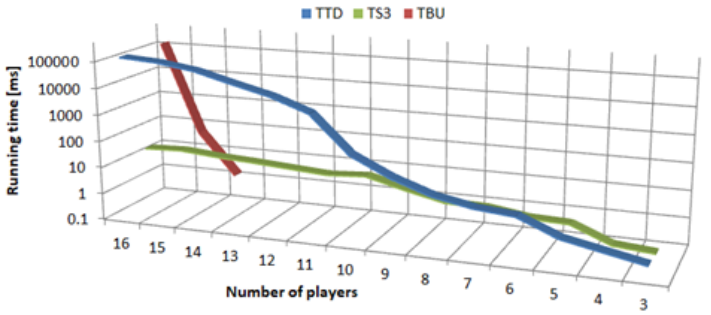
$$Do_s = \frac{Do_o}{Do_1} \tag{17}$$

Utilizing the absolute values of sub-optimal cost Do_0 and optimal cost Do_1, where $Do_1 > Do_0 > 0$. Note that $Do_s = 1$ only when $Do_1 = Do_0$, indicating that the algorithm producing the sub-optimal cost Do_0 is, in fact, optimal. It is important to note that the computation of the relative sub-optimal cost assumes that the exact value Do_1 of the optimal cost is known. In our specific case, this value is known as it was determined using the $OptimalDPTD$ optimal algorithm for $P = 3, 4 \ldots 16$ players.

In Figure 7b, we present a comparison of the running times for algorithms $OptimalDPTD$, $OptimalDPBU$, and $OptimalSubS3$ denoted as TTD, TBU, and $TS3$, respectively. The comparison is made considering $P = 3, 4, \ldots, 16$ players. The running times were obtained by executing each algorithm 13 times using the same input data and plotted on a logarithmic scale. Notably, algorithm $OptimalSubS3$ stands out as the most efficient among the three. The linear increase observed in TTD and TBU on the logarithmic scale confirms our findings, signifying that the complexities of algorithms $OptimalDPTD$ and $OptimalDPBU$ grow exponentially with the number of competition players.



(a)



(b)

Figure 7. (a)Comparison of the relative maximum and average costs achieved by algorithm SubOptS3 ($CoMAX_3$ and $COSa_3$) with the optimal cost attained by algorithm OptimalSubD1 for $P = 3, 4, \ldots, 16$ players, based on our input dataset. (b) We compared the running times of algorithms OptimalDPTD, OptimalDPBU, and OptimalSubS3 for $P = 3, 4, \ldots, 16$ players, using our input dataset. The time values are displayed on a logarithmic scale.

The same tendency is also noticeable in the plot of TBU, where the values were recorded only for an exact power of two of the number of players, i.e., $P = 4, 8, 16$. Furthermore, the sub-linear increasing trend of TS3 on the logarithmic scale aligns with the polynomial time complexity of algorithm $OptimalSubS3$.

The same tendency is also noticeable in the plot of TBU, where the values were recorded only for an exact power of two of the number of players, i.e., $P = 4, 8, 16$. Furthermore, the sub-linear increasing trend of $TS3$ on the logarithmic scale aligns with the polynomial time complexity of algorithm $OptimalSubS3$.

This study's findings appear to align with previous research in the field of Fixing Knockout Tournaments With Seeds. Several prior studies have also demonstrated a positive correlation between the effective utilization of this tournament format to maximize the number of knockout matches within the context of UEFA Champions League matches. The design was coined "Choose Your Opponent" by the author. For instance, the optimization of the tournament design using utility functions optimized with simulated annealing algorithms and optimal Bayesian design carried out by Hennessy [11]. These consistent findings, in conjunction with our study's results, indicate that the differences in relative cost differences resulting from various sub-algorithms with the same input were rather low.

4. CONCLUSION

The variance in relative costs among different sub-algorithms using the same input was relatively low. This outcome was anticipated since the cost values were generated as integers within a narrow range of $1 \leq 9$, while the costs typically reached much higher values. For instance, when considering a scenario with 25 players (P = 25), the observations revealed that the relative difference between the lowest and highest costs (307.47 and 333.39) was only 2.76%. In summary, among the sub-optimal algorithms compared, the $SubOptS_3$ algorithm yielded the best results. Consequently, our optimized single-elimination system can effectively apply to sports competitions employing a bye system. Potential areas for further research in this study are exploring techniques

to parallelize dynamic programming algorithms to speed up the computation of large knockout tournaments taking advantage of multi-core processors or distributed computing environments.

5. ACKNOWLEDGEMENTS

This research is supported by LPPM of Universitas Islam Kalimantan Muhammad Arsyad Al Banjari Banjarmasin (UNISKA MAB).

6. DECLARATIONS

AUTHOR CONTIBUTION

The first author conceptualized this study, conducted experiments, wrote the original draft, and revised the manuscript. The first author wrote the manuscript and performed the experiments. AR made the experimental plan, supervised the work, and revised the manuscript. The third author performed the data analysis and revised the manuscript. The first author made the experimental plan and revised the manuscript. The second author evaluated the developed technique and revised the manuscript. The third author designed the experimental plan, supervised the work, and revised the manuscript. All authors have read and agreed to the published version of the manuscript.

FUNDING STATEMENT

This work was supported by LP2M UNISKA MAB (APBU Ta.2022/2023)

COMPETING INTEREST

The authors declare that the research was conducted without any commercial or financial relationships that could be construed as a potential conflict of interest.

REFERENCES

[1] P. Manurangsi and W. Suksompong, "Fixing Knockout Tournaments With Seeds," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 412–418, 2022.

[2] M. P. Kim, W. Suksompong, and V. V. Williams, "Who Can Win a Single-Elimination Tournament?" pp. 1–13, 2018.

[3] L. Csató, "A simulation comparison of tournament designs for the World Men's Handball Championships," *International Transactions in Operational Research*, vol. 28, no. 5, pp. 2377–2401, 2021.

[4] E. C. Olymphic, "New Study on the Economic impact of Sport released by the European Commission," 2020.

[5] R. Hulett, "Single-elimination brackets fail to approximate Copeland winner," *Leibniz International Proceedings in Informatics, LIPIcs*, vol. 145, no. 13, pp. 1–13, 2019.

[6] Y. Ikhwani, K. Marzuki, and A. Ramadhan, "Automated University Lecture Schedule Generator based on Evolutionary Algorithm," vol. 22, no. 1, pp. 127–136, 2022.

[7] J. Kuo and J. W. Sheppard, "Tournament Topology Particle Swarm Optimization," *2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings*, pp. 2265–2272, 2021.

[8] F. Spieksma, R. Pendavingh, and R. Lambers, "How to Design a Stable Serial Knockout Competition," 2022.

[9] K. Gokcesu and H. Gokcesu, "Merging Knockout and Round-Robin Tournaments: A Flexible Linear Elimination Tournament Design," pp. 1–6, 2022.

[10] D. Bădică, A., Bădică, C., Buligiu, I., Ciora, L.I., Logoftu, "Optimal Knockout Tournaments: Definition and Computation," in *International Conference on Large-Scale Scientific Computing*, 2022.

[11] F. Della Croce, G. Dragotto, and R. Scatamacchia, "On fairness and diversification in WTA and ATP tennis tournaments generation," *Annals of Operations Research*, vol. 316, no. 2, pp. 1107–1119, 2022.

[12] J. Guyon, "Choose your opponent: A new knockout design for hybrid tournaments," *Journal of Sports Analytics*, vol. 8, no. 1, pp. 9–29, 2021.

[13] H. Aziz, S. Gaspers, S. Mackenzie, N. Mattei, P. Stursberg, and T. Walsh, "Fixing balanced knockout and double elimination tournaments," *Artificial Intelligence*, vol. 262, no. May, pp. 1–14, 2018.

[14] A. Karpov, "Generalized knockout tournament seedings," *International Journal of Computer Science in Sport*, vol. 17, no. 2, pp. 113–127, 2018.

[15] P. Bhumipol, N. Makaje, T. Kawjaratwilai, and R. Ruangthai, "Match analysis of professional Muay Thai fighter between winner and loser," *Journal of Human Sport and Exercise*, vol. 18, no. 3, pp. 657–669, 2023.

[16] B. R. Sziklai, P. Biró, and L. Csató, "The efficacy of tournament designs," *Computers and Operations Research*, vol. 144, no. February, 2022.

[17] T. Stojadinović, "On catalan numbers," *Teaching of Mathematics*, vol. 18, no. 1, pp. 16–24, 2015.

[18] O. Banias, "Test case selection-prioritization approach based on memoization dynamic programming algorithm," *Information and Software Technology*, vol. 115, no. June, pp. 119–130, 2019.

[19] A. Bădică, C. Bădică, I. Buligiu, L. I. Ciora, and D. Logoftu, "Dynamic programming algorithms for computing optimal knockout tournaments," *Mathematics*, vol. 9, no. 19, 2021.

[20] D. Knuth, *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms*, part 1 ed. Boston: Addison-Wesley Professional, 2011.

[This page intentionally left blank.]