Multivariate Analysis of Scheduling Fair Competitions

Siddharth Gupta
Ben-Gurion University of the Negev
BeerSheva, Israel
siddhart@post.bgu.ac.il

Meirav Zehavi
Ben-Gurion University of the Negev
BeerSheva, Israel
meiravze@bgu.ac.il

ABSTRACT

A *fair competition*, based on the concept of envy-freeness, is a non-eliminating competition where each contestant (team or individual player) may not play against all other contestants, but the total difficulty for each contestant is the same: the sum of the initial rankings of the opponents for each contestant is the same. Similar to other non-eliminating competitions like the Round-robin competition or the Swiss-system competition, the winner of the fair competition is the contestant who wins the most games. The Fair Non-Eliminating Tournament (Fair-NET) problem can be used to schedule fair competitions whose infrastructure is known. In the Fair-NET problem, we are given an infrastructure of a tournament represented by a graph G and the initial rankings of the contestants represented by a multiset of integers S. The objective is to decide whether G is *S-fair*, i.e., there exists an assignment of the contestants to the vertices of G such that the sum of the rankings of the neighbors of each contestant in G is the same constant $k \in \mathbb{N}$. We initiate a study of the classical and parameterized complexity of Fair-NET with respect to several central structural parameters motivated by real world scenarios, thereby presenting a comprehensive picture of it.

KEYWORDS

Tournament Scheduling; Fixed Parameter Tractable; Fair Allocation; Computational Social Choice

ACM Reference Format:
Siddharth Gupta and Meirav Zehavi. 2021. Multivariate Analysis of Scheduling Fair Competitions. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 10 pages.

1 INTRODUCTION

Various real life situations require to conduct fair competitions. For illustration, suppose we want to schedule a non-eliminating sports competition in which there are n contestants and n grounds located on the circumference of a circle. As organizers, we want to assign a home ground to each contestant in such a way that every contestant c plays only against r contestants whose home ground is nearest to c's home ground rather than all the contestants. The underlying rationale can be time constraints and also to minimize the travel time for each contestant (similar to the Traveling Tournament problem, see e.g. [22, 23]). However, the total difficulty for each contestant should be the same, i.e., the sum of the initial rankings of the opponents for each contestant is the same. We can model this problem as an instance of the Fair Non-Eliminating Tournament (Fair-NET) problem, where we are given an infrastructure of a tournament represented by a graph G and the initial rankings of the contestants represented by a multiset of integers S. The objective is to decide whether G is *S-fair*, i.e., there exists an assignment of the contestants to the vertices of G such that the sum of the rankings of the neighbors of each contestant in G is the same constant $k \in \mathbb{N}$. Here, k is called the *S-fairness constant*, or simply *fairness constant* if S is clear from the context, of G. Clearly, the above problem is equivalent to having an r-regular graph G with n vertices, one for each ground, and edges connecting each vertex to $r/2$ nearest vertices on the left and $r/2$ nearest vertices on the right, and the objective is to determine whether G is S-fair where S is the multiset of the rankings of the contestants (see Figure 1). As the total difficulty for each contestant in the competition is the same, we refer to such a competition as a *fair competition*.

In general, if we have the infrastructure of the competition (implicitly, like the above example, or explicitly) and we want to schedule a fair competition, we can model the problem as that of determining whether the graph representing the infrastructure of the competition is S-fair where S is the multiset of the rankings of the contestants. This situation is very frequently observed in on-line games or in other recurring competitions - in such competitions, the infrastructure of the competition is fixed and the set of contestants keeps changing.

Scheduling competitions and tournaments with different objectives is a well studied problem in the literature. There are, mainly, two fundamental competition designs, with all other designs considered as variations and hybrids. The first one is the elimination (or knockout) competition, in which the contestants are mapped to the leaf nodes of a complete binary tree. Contestants mapped to nodes with same parent compete against each other in a match, and the winner of the match moves up the tree. The contestant who reaches the root node is the winner of the tournament. The second one is the non-eliminating competition, in which no contestant is eliminated after one or few loses, and the winner is decided at the end of all the games by selecting the contestant with largest number of wins.

In recent years, algorithmic perspectives of scheduling both kinds of competitions have received significant attention by the computational social choice community. We will first discuss elimination competitions, followed by non-eliminating competitions. With respect to elimination competitions, the design of a fair elimination competition under various definitions of being fair has received notable attention [24, 39, 47, 48]. In this context, it is also relevant to mention the Tournament Fixing problem. Here, we are given n contestants, an encoding of the outcome of each potential match between every two contestants as a digraph D, and a favorite contestant v: the goal is to design an elimination tournament so that v wins the tournament. This problem was introduced

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

by Vu *et al.* [46]. After this, it was extensively studied from both combinatorial and algorithmic (as well as parameterized) points of view [3, 20, 28–30, 35, 41, 42, 50].

With respect to non-eliminating competitions, a round-robin tournament (RRT) is one of the most popular forms, in which each contestant plays every other contestant [38]. A well studied problem regarding RRTs is the Traveling Tournament problem, where the goal is to design a fair RRT by minimizing the total travel distance for every team [19, 22, 23, 31, 44, 51]. Another related problem is to design a fair RRT by minimizing the number of "breaks" during the tournament [36, 45, 52]. Despite of being a popular non-eliminating competition, RRTs have some disadvantages. The first disadvantage of this format is the long tournament length, as each contestant plays against all other contestants. From the fairness point of view, a second disadvantage of this format, also mentioned in [38], is that it favors the *strongest* contestants (i.e., the contestants with the highest initial ranking). To see this, let $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ be the initial rankings of the n contestants, where r_i is the initial ranking of contestant i, and let R be the total sum of the rankings. In RRT, the total difficulty faced by contestant i is $R - r_i$, which shows that the total difficulty faced by a contestant increases as we go from the strongest contestants to the weakest contestants. In light of the above disadvantage, motivated by one of the definitions proposed for fairness in [38, 39, 48], and based on a popular fairness concept called *envy-freeness* introduced by Foley [13] in the study of fair division and allocation problems in multi-agent systems (see, e.g. [5, 7, 8, 17]), we define a *fair competition* in an attempt to address both the above disadvantages with RRTs. Here, a *fair competition* is one where each contestant plays with a *subset* of all other contestants, yet the total difficulty for each contestant in the competition is the same. Similar to an envy-free division where no agent feels envy of another agent's share, in a fair competition no contestant feels envy about another contestant's schedule as the total difficulty for each contestant is the same.

Apart from scheduling fair competitions, Fair-NET can be used to model other computational problems in social choice. For example, suppose we have m candidates and n jobs, and every job is associated with an integer "reward". Every candidate can choose r jobs and every job is chosen by exactly one candidate. Now, we want to get an assignment of the jobs to the candidates such that the total reward collected by every candidate is k, for some integer k. Then, this is equivalent of having a graph G that is a collection of m stars, each having r leaves, with n total leaves, and the objective is to determine whether G is S-fair with the fairness constant k where S is the union of (i) the multiset of rewards and (ii) the multiset $S' = \{k, \ldots, k\}$ containing the element k m times. Intuitively, S' represents the multiset of rewards collected by every candidate. Every star vertex corresponds to a candidate c, and its leaves correspond to the jobs c is assigned to.

The Fair-NET problem can also be used to design semi-magic and magic squares [49] defined as follows. A *semi-magic square* is an $n \times n$ grid ($n \geq 3$) filled with positive integers from a multiset I such that each cell contains a distinct integer occurrence in I and the sum of integers in each row and each column is the same. A *magic square* is a semi-magic square with the additional constraint that the sum of the integers in both the diagonals is also the same and equal to the sum of integers in each row and each column. The details about how to model a semi-magic square (and similarly a magic square) as an instance of Fair-NET can be found in [21].

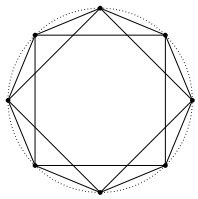

Figure 1: Example of a 4 regular graph where every vertex is connected to 2 vertices on the left and 2 on the right.

Table 1: Summary of our results. Here Δ, tw, fvs and vc denote the maximum degree, treewidth, feedback vertex set number and vertex cover number of the input graph, respectively; α denotes the number of distinct elements in S. Note that $\texttt{tw} \leq \texttt{fvs} \leq \texttt{vc}$.

Parameters	*Parameterized Complexity*
$\texttt{tw} + \Delta$	NP-hard for $\texttt{tw} = 3, \Delta = 3$ (also for regular graphs) [Theorem 3.1]
$\alpha + \Delta$	NP-hard for $\alpha = 3, \Delta = 6$ (also for regular graphs) [Theorem 3.3]
$\texttt{fvs} + \Delta$	NP-hard for $\texttt{fvs} = 0, \Delta = 3$ [Theorem 3.2]
$\texttt{fvs} + \Delta + \alpha$	FPT [Theorem 4.6]
fvs	FPT (for regular graphs) [Theorem 4.8]
$\texttt{vc} + \alpha$	FPT [Theorem 4.7]

1.1 Our Contribution and Methods

To the best of our knowledge, while Fair-NET has been studied extensively from a combinatorial point of view (discussed later in the section), close to nothing is known from an algorithmic point of view. We initiate a systematic algorithmic study of Fair-NET. On the one hand, we show NP-hardness results on special graph classes, which imply para-NP-hardness for the problem with respect to several combinations of structural graph parameters. (For basic notions in parameterized complexity, see Section 2). On the other hand, we show that the problem is *fixed-parameter tractable (FPT)* for four different combinations of these parameters.

The choice of our parameters is motivated by the real world examples from the introduction. In the example of fair competition, we may want every contestant to play only a fraction of the total possible games, which in turn means that the maximum degree Δ of the infrastructure graph is small compared to the total number of contestants. Similarly, it is likely to happen that a lot of contestants have the same rankings or a lot of jobs have the same rewards, which implies that the number α of unique elements in S is small compared to the total number of contestants or jobs. In the case of job assignment, the underlying graph is a set of stars, which has treewidth 1 and the size of minimum feedback vertex set is 0. Moreover, treewidth, feedback vertex set and vertex cover are central parameters in the field of parameterized complexity.

Our main results are as follows (summarized in Table 1). First, we show that Fair-NET is NP-hard for three different graph classes: disjoint unions of $K_{3,3}$'s, disjoint unions of $K_{1,3}$'s and 6-regular graphs with 3 distinct labels. Consequently, it is para-NP-hard parameterized by (i) treewidth plus maximum degree, (ii) maximum degree plus feedback vertex set number, and (iii) maximum degree plus the number of distinct labels. The para-NP-hard results hold even for regular graphs when parameterized by either treewidth plus maximum degree or maximum degree plus the number of distinct labels.

Second, we show that Fair-NET is FPT parameterized by (i) maximum degree plus feedback vertex set number plus the number of distinct labels, (ii) vertex cover number plus the number of distinct labels, and (iii) feedback vertex set number for regular graphs. We derive some of these results by using insights into Fair-NET itself when the input graph is a cycle, a disjoint union of stars, or a connected graph with minimum degree 1, and Integer Linear Programming.

Our choice of parameters also shows several borders of (in-)tractability. For example, the problem is para-NP-hard when parameterized by either $\Delta + \alpha$ or by $\mathtt{fvs} + \Delta$, but becomes FPT when parameterized by $\mathtt{fvs} + \Delta + \alpha$. Similarly, it is para-NP-hard by $\mathtt{fvs} + \Delta$, but becomes FPT by $\mathtt{fvs}$ for regular graphs. Overall, we give a comprehensive picture of the classical and parameterized complexity of Fair-NET. For lack of space, some results and proofs marked with an asterisk ($*$) are omitted or sketched; for completeness see [21].

Related Work. The Fair-NET problem was first introduced by Vilfred [26] when $S = \{1, 2, \ldots, n\}$. Such a labeling is called *sigma-labeling* in that paper. The concept of fair scheduling was independently studied by Miller *et al.* [32] in 2003 under the name 1-*vertex magic* and by Sugeng *et al.* [43] under the name *distance magic labeling*. For recent surveys on distance magic labeling, see [1, 37]. The Fair-NET problem for a general multiset S was first studied by O'Neal and Slater [33]. In the same paper, they also proved that if a graph G is S-fair, then the S-fairness constant of G is unique. In [40], Slater proved that Fair-NET is NP-hard. More recently, Godinho *et al.* [18] studied the special case of Fair-NET where S is a set and not a multiset. They gave a simpler proof for the uniqueness of S-fairness constant and also exhibited several families of S-fair graphs. Recently, the same set of authors studied a measure called *distance magic index* related to S-fair labeling and determined the distance magic index of trees and complete bipartite graphs in [2]. There has also been a long line of studies on other kinds of graph labeling, like *{0,1}*-Fair-NET, where we consider the closed neighborhood of every vertex instead of open neighborhood (i.e., the vertex itself is also considered in its neighbor set). Another example is *vertex-bimagic labeling*, in which there exists two constants k_1 and k_2 such that the sum of neighbors of every vertex is either k_1 or k_2. For more information, see the recent survey [15].

2 PRELIMINARIES

Sets and Functions. Given two multisets $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$, their *disjoint union* is the multiset $S = A \uplus B = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$. For example, let $A = \{1, 3, 4, 5, 5\}$ and $B = \{3, 2, 4, 6\}$. Then, $A \uplus B = \{1, 2, 3, 3, 4, 4, 5, 5, 6\}$. Given a multiset S, $\alpha(S)$ denotes the number of distinct elements in S, and for every $a \in S$, $\alpha_S(a)$ denotes the number of times a appear in S. Given a multiset A, $\sum A$ denotes the sum of its elements (in case they are integers), and $|A|$ denotes its size. For any $t \in \mathbb{N}$, $[t]$ denotes the set $\{1, 2, \ldots, t\}$. Given a function f defined on a multiset A, $f(A) = \{f(a) : a \in A\}$. Let $f : A \to B$ be a function from a multiset A to a multiset B. Then the *restriction of* f to a multiset $A' \subseteq A$ is the function $f|_{A'} : A' \to B$ given as $f|_{A'}(x) = f(x)$ for every $x \in A'$.

Graphs. In this paper, we consider only undirected graphs. Given a graph G, we denote its vertex set and edge set by $V(G)$ and $E(G)$, respectively. For a vertex $v \in V(G)$, the set of all the neighbors of v in G is denoted by $N_G(v)$, i.e. $N_G(v) = \{u \in V(G) \mid \{u, v\} \in E(G)\}$. The degree of a vertex $v \in V(G)$ in G is denoted by $\deg_G(v)$. When G is clear from the context, we drop the subscript. Given an induced subgraph H of G, the set of neighbors of vertices in H which are not in H is denoted by $N_G(H)$, i.e., $N_G(H) = \left(\bigcup_{v \in V(H)} N_G(v)\right) \setminus V(H)$. The maximum and minimum degree of G are denoted by $\Delta(G)$ and $\delta(G)$, respectively. Given a set $V' \subseteq V(G)$, the subgraph of G induced by V' is denoted by $G[V']$. A path on n vertices is denoted by P_n. A cycle on n vertices is denoted by C_n. A complete bipartite graph with bipartition A and B such that $|A| = m, |B| = n$ is denoted by $K_{m,n}(A, B)$. If A and B are clear from the context, we write $K_{m,n}(A, B)$ as $K_{m,n}$. Given a forest F, the set of leaves of F is denoted by leaves(F). Given a rooted tree T, for a vertex $v \in V(T)$, the set of children of v in T is denoted by children$_T(v)$.

An *r-regular graph* is a regular graph where every vertex has degree r. An *n-star* graph (on $n+1$-vertices) is the complete bipartite graph $K_{1,n}$. Given an n-star graph where $n \geq 2$, the *star-vertex* is the unique vertex with degree n. The *disjoint union* of two graphs G_1 and G_2, denoted by $G_1 + G_2$, is the graph with vertex set $V(G_1) \uplus V(G_2)$ and edge set $E(G_1) \uplus E(G_2)$. For any $m \in \mathbb{N}$, we denote the disjoint union of m copies of a graph G by mG. Note that, the disjoint union of two or more nonempty graphs is always a disconnected graph. For other standard notations not explicitly defined here, we refer to the book [11].

The *treewidth, vertex cover number* and *feedback vertex set number* of a graph G are defined as follows.

*Definition 2.1 (***Treewidth***).* A *tree decomposition* of a graph G is a tree T whose nodes, called *bags*, are labeled by subsets of vertices of G. For each vertex v, the bags containing v must form a nonempty contiguous subtree of T, and for each edge $\{u, v\}$, at least one bag must contain both u and v. The *width* of the decomposition is one less than the maximum cardinality of any bag, and the *treewidth* $\mathtt{tw}(G)$ of G is the minimum width of any of its tree decompositions.

Based on the definition of treewidth, we have the following observation about disjoint union.

Observation 1. *The treewidth of the disjoint union of two vertex-disjoint graphs G_1 and G_2 is max$\{\mathtt{tw}(G_1), \mathtt{tw}(G_2)\}$.*

*Definition 2.2 (***Vertex Cover***).* A *vertex cover* of a graph G is a set of vertices in G such that every edge in G has at least one endpoint in the set. We denote the minimum size of a vertex cover of G by $\mathtt{vc}(G)$.

Definition 2.3 (**Feedback Vertex Set**). A *feedback vertex set* of a graph G is a set of vertices whose removal results in an acyclic graph. We denote the minimum size of a feedback vertex set of G by $\texttt{fvs}(G)$.

We will show hardness results from 3-Partition and a variant of SAT called 3-XSAT$^3_+$ (which were proved to be strongly NP-complete and NP-complete in [16] and [34], respectively), defined as follows.

Definition 2.4 (**3-Partition**). Given a multiset W of $n = 3m$ positive integers, for some $m \in \mathbb{N}$, can W be partitioned into m triplets $W_1, W_2, \ldots, W_m$ (i.e., $W = \biguplus_{i\in[m]} W_i$ and $|W_i| = 3$ for every $i \in [m]$) such that for every $i \in [m]$, $\sum W_i = \sum W/m$?

Definition 2.5 (**3-XSAT$^3_+$**). Given a formula in conjunctive normal form (CNF) where all literals are positive, each clause has size exactly 3, and each variable occurs exactly 3 times, does there exist a truth assignment to the variables so that each clause has exactly one true variable?

Fair Non-Eliminating Tournament. Given an infrastructure of a tournament represented by a graph G and the initial rankings of the contestants represented by a multiset of integers S, G is called *S-fair* if there exists an assignment of the contestants to the vertices of G such that the sum of the rankings of the neighbors of each contestant in G is the same. Equivalently, given the infrastructure graph G and the multiset of contestants' rankings S with $|S| = |V(G)|$, G is *S-fair* if there exists a bijection $f : V(G) \to S$ such that for every vertex $v \in V(G)$, $\sum f(N(v)) = k$, where k is a constant called *S-fairness constant*.

For any vertex $v \in V(G)$, $f(v)$ is called the *label* of v. We denote the set of all bijective functions that satisfy the above property by $\mathcal{M}(G, S)$. In [33], O'Neal and Slater showed that if a graph G is S-fair, then its S-fairness constant is unique.

Since the infrastructure graph of a tournament is an undirected graph and the initial ranking of several players can be the same, we define the Fair Non-Eliminating Tournament (Fair-NET) problem as follows.

Definition 2.6 (**Fair-NET Problem**). Given an undirected graph G and a multiset of positive integers S with $|S| = |V(G)|$, is G S-fair (i.e. $\mathcal{M}(G, S) \neq \emptyset$)?

The following observations about S-fair graphs follow directly from its definition.

Observation 2 (**Label Swap**). *Let G be an S-fair graph. Let $f \in \mathcal{M}(G, S)$. Let $u, v \in V(G)$ such that $N_G(u) = N_G(v)$. Consider $f' : V(G) \to S$, defined as follows. For all $w \in V(G) \setminus \{u, v\}$, $f'(w) = f(w)$; $f'(u) = f(v)$; $f'(v) = f(u)$. Then $f' \in \mathcal{M}(G, S)$.*

Observation 3. *Given two graphs G_1 and G_2 and a multiset of positive integers S, $G_1 + G_2$ is S-fair if and only if G_1 is S_1-fair and G_2 is S_2-fair with the same fairness constant for some $S_1, S_2 \subseteq S$ such that $S_1 \uplus S_2 = S$.*

Observation 4. *Given a complete bipartite graph $K_{m,n}(A, B)$ and a multiset of positive integers S, $K_{m,n}$ is S-fair if and only if there exists a bijection $f : V(K_{m,n}) \to S$ such that $\sum f(A) = \sum f(B) = \sum S/2$.*

Integer Linear Programming. In the Integer Linear Programming Feasibility (ILP) problem, the input consists of p variables $x_1, x_2, \ldots, x_p$ and a set of m inequalities of the following form:

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_1 + \cdots + a_{1,p}x_p &\leq y_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,p}x_p &\leq y_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,p}x_p &\leq y_m \end{aligned}$$

where every coefficient a_{i_j} and y_i is required to be an integer. The task is to check whether there exists an assignment of integer values for every variable x_i so that all inequalities are satisfiable. The following theorem about the tractability of the ILP problem will be useful throughout Section 4.

Theorem 2.7 ([14, 25, 27]). *The ILP problem with p variables is FPT parameterized by p.*

Parameterized Complexity. A problem Π is a *parameterized* problem if each problem instance of Π is associated with a *parameter* k. For simplicity, we denote a problem instance of a parameterized problem Π as a pair (I, k) where the second argument is the parameter k associated with I. The main objective of the framework of Parameterized Complexity is to confine the combinatorial explosion in the running time of an algorithm for an NP-hard parameterized problem Π to depend only on k. In particular, a parameterized problem Π is *fixed-parameter tractable* (FPT) if any instance (I, k) of Π is solvable in time $f(k) \cdot |I|^{O(1)}$, where f is an arbitrary computable function of k. Moreover, a parameterized problem Π is *para-NP-hard* if it is NP-hard for some fixed constant value of the parameter k. For more information on Parameterized Complexity, we refer the reader to books such as [10, 12].

3 PARA-NP-HARDNESS RESULTS

In this section, we exhibit the para-NP-hardness of Fair-NET with respect to several structural graph parameters. We start with a para-NP-hardness result with respect to the parameter $\texttt{tw} + \Delta$.

Theorem 3.1. *The* Fair-NET *problem is NP-hard for* 3*-regular graphs with treewidth* 3*. In particular, it is para-NP-hard parameterized by* $\texttt{tw} + \Delta$*, even for regular graphs.*

Proof. We present a reduction from 3-Partition. Given a multiset W of $n = 3m$ positive integers, for some $m \in \mathbb{N}$, we create two instances of Fair-NET based on the value of m as follows (see Figure 2).

Case 1 [When m is a multiple of 2]: In this case, we create an instance (G, S) of Fair-NET where $G = (m/2)K_{3,3}$ and $S = W$. Note that G is a 3-regular graph. Since $\texttt{tw}(K_{3,3}) = 3$, by Observation 1, $\texttt{tw}(G) = 3$. Let $V(G) = \biguplus_{i\in[m/2]} V_i$ where $V_i = A_i \cup B_i$ is the vertex set of the i-th copy of $K_{3,3}$ with bipartition A_i and B_i. We now prove that W is a Yes instance of 3-Partition if and only if G is S-fair.

Assume first that W is a Yes instance of 3-Partition. Let $W_1, W_2, \ldots, W_m$ be a corresponding partition of W. Then, by Definition 2.4, for every $i \in [m]$, $\sum W_i = \sum W/m$. Let $f : V(G) \to S$ be a bijective function defined as follows. For every $i \in [m/2]$, let $f(A_i) = W_i$ and $f(B_i) = W_{m/2+i}$ (the internal labeling within A_i and B_i is arbitrary).

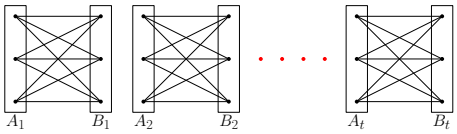

Figure 2: Example of the graph G built in the reduction of Theorem 3.1. $t = m/2$ for Case 1 and $t = (m+1)/2$ for Case 2.

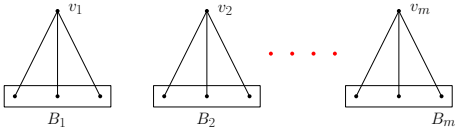

Figure 3: Example of the graph G built in the reduction of Theorem 3.2.

So, for every $i \in [m/2], \sum f(A_i) = \sum f(B_i) = \sum W/m$. Thus, by Observations 3 and 4, $G = (m/2)K_{3,3}$ is S-fair.

Conversely, let $G = (m/2)K_{3,3}$ be S-fair. Then, by Observations 3 and 4, there exists a bijection $f : V(G) \to S$ such that for every $i \in [m/2], \sum f(A_i) = \sum f(B_i) = \sum S/m = \sum W/m$. Thus, $\{f(A_1), f(A_2), \ldots, f(A_{m/2}), f(B_1), \ldots, f(B_{m/2})\}$ is a partition of W satisfying the required property, so W is a Yes instance of 3-Partition.

Case 2 [When m is not a multiple of 2]: Without loss of generality, we can assume that every element in W is greater than 1 as otherwise we can get an equivalent instance of 3-Partition by adding 1 to all the elements of W. Let $sum = \sum W/m$ be the required sum of every subset. As every element in W is greater than 1, $sum \geq 6$. In this case, we create an instance (G, S) of Fair-NET where $G = \big((m+1)/2\big)K_{3,3}$ and $S = W \uplus \{sum - 2, 1, 1\}$. Note that G is a 3-regular graph and $\mathtt{tw}(G) = 3$. Let $V(G) = \biguplus_{i \in [m+1/2]} V_i$ where $V_i = A_i \cup B_i$ is the vertex set of the i-th copy of $K_{3,3}$ with bipartition A_i and B_i. We now prove that W is a Yes instance of 3-Partition if and only if G is S-fair.

Assume first that W is a Yes instance of 3-Partition. Let $W_1, W_2, \ldots, W_m$ be the corresponding partition of W. Then, by Definition 2.4, for every $i \in [m], \sum W_i = \sum W/m = sum$. Let $W_{m+1} = \{sum - 2, 1, 1\}$. Clearly, $\sum W_{m+1} = sum$. As $S = W \uplus \{sum - 2, 1, 1\}$, $S = \biguplus_{i \in [m+1]} W_i$. Let $f : V(G) \to S$ be a bijective function defined as follows. For every $i \in [(m+1)/2]$, let $f(A_i) = W_i$ and $f(B_i) = W_{(m+1)/2+i}$ (the internal labeling within A_i and B_i is arbitrary). So, for every $i \in [(m+1)/2], \sum f(A_i) = \sum f(B_i) = sum$. Thus, by Observations 3 and 4, $G = \big((m+1)/2\big)K_{3,3}$ is S-fair.

Conversely, let $G = (m+1)/2K_{3,3}$ be S-fair. Then, by Observations 3 and 4, there exists a bijection $f : V(G) \to S$ such that for every $i \in [(m+1)/2], \sum f(A_i) = \sum f(B_i) = \sum S/(m+1) = \sum W/m$. Without loss of generality, let A_s be the set containing $sum - 2$, for some $s \in [(m+1)/2]$. As $\sum A_s = sum$, $|A_s| = 3$ and all the elements in W are greater than 1, necessarily $A_s = \{sum - 2, 1, 1\}$. As $S = W \uplus \{sum - 2, 1, 1\}$, we get that $\{f(A_1), \ldots, f(A_{s-1}), f(A_{s+1}), \ldots, f(A_{(m+1)/2}), f(B_1), \ldots, f(B_{(m+1)/2})\}$ is a partition of W satisfying the required property, so W is a Yes instance of 3-Partition. □

We now proceed with the para-NP-hardness result with parameter $\mathtt{fvs} + \Delta$.

Theorem 3.2. *The* Fair-NET *problem is NP-hard for forests with* $\Delta = 3$. *Since forests have* $\mathtt{fvs} = 0$, Fair-NET *is para-NP-hard parameterized by* $\mathtt{fvs} + \Delta$.

Proof. We present a simple reduction from 3-Partition. Given a multiset W of $n = 3m$ positive integers, for some $m \in \mathbb{N}$, let $sum = \sum W/m$ be the required sum of every subset. We create an instances (G, S) of Fair-NET where $G = mK_{1,3}$ and $S = W \uplus \{s_1 = sum, s_2 = sum, \ldots, s_m = sum\}$. Note that G is a forest with $\Delta(G) = 3$. Let $V(G) = \biguplus_{i \in [m]} V_i$ where $V_i = \{v_i\} \cup B_i$ is the vertex set of the i-th copy of $K_{1,3}$ with B_i being the set of leaves. See Figure 3. We now prove that if W is a Yes instance of 3-Partition if and only if G is S-fair.

Assume first that W is a Yes instance of 3-Partition. Let $W_1, W_2, \ldots, W_m$ be the corresponding partition of W. Then, by Definition 2.4, for every $i \in [m], \sum W_i = sum$. Let $f : V(G) \to S$ be a bijective function defined as follows. For every $i \in [m]$, let $f(B_i) = W_i$ and $f(v_i) = sum$ (the internal labeling of B_i is arbitrary). So, for every $i \in [m], f(v_i) = \sum f(B_i) = sum$. Thus, by Observations 3 and 4, $G = mK_{1,3}$ is S-fair.

Conversely, let $G = mK_{1,3}$ be S-fair. Then, by Observations 3 and 4, there exists a bijection $f : V(G) \to S$ such that for every $i \in [m], f(v_i) = \sum f(B_i) = \sum S/(m+1) = sum$. So, we get that $\{f(B_1), \ldots, f(B_m)\}$ is a partition of W satisfying the required property, so W is a Yes instance of 3-Partition. □

Finally, we give the para-NP-hardness result with parameter $\alpha + \Delta$.

Theorem 3.3 (*). *The* Fair-NET *problem in NP-hard for 6-regular graphs with 3 distinct labels. In particular, it is para-NP-hard parameterized by* $\alpha + \Delta$, *even for regular graphs.*

Proof sketch. We present a reduction from 3-XSAT$^3_+$. Given a 3-XSAT$^3_+$ formula ρ with n variables and n clauses, we create an instance (G, S) of Fair-NET as follows. Suppose that the variables are indexed by $1, 2, \ldots, n$ and so do the clauses. For every $i \in [n]$, the variable gadget in G consists of a single vertex x_i called a variable vertex. For every $i \in [n]$, the clause gadget in G consists of 15 vertices $c^1_i, c^2_i, \ldots, c^{15}_i$. For every $i \in [n]$, we add the edges among clause vertices and between variable and clause vertices in G as shown in Figure 4. This completes the construction of G. Note that $|V(G)| = n + 15n = 16n$. We now define S as the multiset containing 3 distinct labels 1, 2 and 4 with $\alpha_S(1) = 2n/3, \alpha_S(2) = 15n$ and $\alpha_S(4) = n/3$. From the above construction, it is easy to see that G is a 6-regular graph and S contains 3 distinct labels. We set a variable to true if and only if the label of the corresponding variable vertex is 4 and false otherwise. In [21], we prove that ρ is satisfiable if and only if G is S-fair. □

4 FPT ALGORITHMS

In this section, we develop FPT algorithms for the Fair-NET problem with respect to several structural graph parameters. We begin by giving the following observation for a graph G to be S-fair when G contains isolated vertices. If G contains isolated vertices (i.e., $\delta(G) = 0$), then G is S-fair if and only if for every vertex $v \in V(G), \deg_G(v) = 0$ (i.e., G contains only isolated vertices). Due

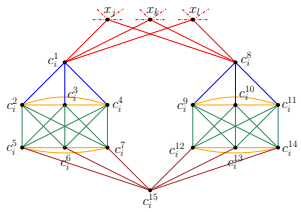

Figure 4: Example of the clause gadget in G for a clause $c_i = x_j \vee x_k \vee x_l$ built in the reduction of Theorem 3.3.

to this observation, in the rest of this section, we assume that G does not contain any isolated vertices. We now give conditions that a graph G must satisfy to be S-fair when G is a cycle or $\delta(G) = 1$.

Lemma 4.1 (*). *Let G be a connected graph with $\delta(G) = 1$ and S be a multiset of positive integers. Then, G is S-fair only if G is a star.*

Lemma 4.2 (*). *Let G be a cycle graph on n vertices and S be a multiset of positive integers. Let k be the required S-fairness constant. Then:*

- *If $n \bmod 4 = 0$, then G is S-fair if and only if S contains 4 labels $a, b, k-a, k-b$ with $\alpha_S(a) = \alpha_S(b) = \alpha_S(k-a) = \alpha_S(k-b) = n/4$, for some $a, b \in \mathbb{N}$ such that $a, b < k$.*
- *If $n \bmod 4 \neq 0$, then G is S-fair if and only if S contains only one label, $k/2$, with $\alpha_S(k/2) = n$.*

We first prove that Fair-NET is FPT parameterized by $\mathtt{fvs}+\alpha+\Delta$. The following two lemmas will be helpful in proving it.

Lemma 4.3. *There exists an $O(|V(G)|)$-time algorithm that, given (i) a graph G, (ii) a multiset of positive integers S, (iii) an induced subgraph F of G such that its a forest, and (iv) a bijection f' from the set $V' = N_G(F) \cup \text{leaves}(F)$ to a subset S' of S, returns another bijection f'' from $N_G(F) \cup V(F)$ to a set S''[1] such that $S' \subseteq S''$ and $f' = f''|_{V'}$. Moreover, if G is S-fair and $f' = f|_{N_G(F)\cup\text{leaves}(F)}$ for some $f \in \mathcal{M}(G, S)$, then $f'' = f|_{N_G(F)\cup V(F)}$.*

Proof. Let (G, k) be an instance of Fair-NET. Let k be the required S-fairness constant. Let $\mathcal{F} = \{T_1, T_2, \ldots, T_t\}$ be the set of connected components of F. For every tree $T \in \mathcal{F}$, we do the following. Let r be an arbitrarily chosen non-leaf vertex of T. Then, consider T as a rooted tree with r as the root vertex. Let d be the depth of the tree T. We partition the vertex set $V(T) = V_1 \cup V_2 \ldots \cup V_d$, such that V_i contains all the vertices of T at depth i. Note that $V_1 = \{r\}$ and $V_d \subseteq \text{leaves}(T)$. Now, consider a vertex $v \neq r$ in T. We partition $N_G(v) = \{p_v\} \cup \big(\text{children}_T(v) \cap \text{leaves}(T)\big) \cup \big(\text{children}_T(v) \setminus \text{leaves}(T)\big) \cup \big(N_G(v) \setminus V(T)\big)$, where p_v is the parent of v in T, $\text{children}_T(v) \cap \text{leaves}(T)$ is the set of children of v in T that are leaves of T, $\text{children}_T(v) \setminus \text{leaves}(T)$ is the set of children of v in T that are non-leaf vertices of T and $N_G(v) \setminus T$ is the set of neighbors of v not in T.

Given f', we define another function f_T on $V'' = V(T) \setminus \text{leaves}(T)$ recursively as follows.

(i) **Base Case:** For all $v \in V''$ such that v has a leaf child, let w be an arbitrarily chosen leaf child of v. Then, $f_T(v) = k - \sum f'(N_G(w) \setminus V(T))$.

(i) **Recursive Step:** For all $v \in V''$ such that v does not have any leaf child, let w be an arbitrarily chosen child of v. Then, $f_T(v) = k - \sum f_T(\text{children}_T(w) \setminus \text{leaves}(T)) - \sum f'(\text{children}_T(w) \cap \text{leaves}(T)) - \sum f'(N_G(w) \setminus T)$.

Note that, if $v \in V_i$, then $\text{children}_T(v) \subseteq V_{i+1}$. So, we compute f_T by processing vertices of T in the order $V_{d-1}, \ldots, V_1$. We now define f'' from $N_G(F) \cup V(F)$ to $S'' = S' \cup f_{T_1} \cup f_{T_2} \ldots f_{T_t}$ as follows.

(i) For every $v \in N_G(F) \cup \text{leaves}(F)$, $f''(v) = f'(v)$.
(i) For every $i \in [t]$ and $v \in V(T_i) \setminus \text{leaves}(T_i)$, $f''(v) = f_{T_i}(v)$.

Clearly, $f' = f''|_{V'}$ and therefore $S' \subseteq S''$. The above recursive procedure visits every vertex of G at most once, so it runs in time $O(|V(G)|)$.

Now, suppose that G is an S-fair graph and $f' = f|_{N_G(F)\cup\text{leaves}(F)}$ for some $f \in \mathcal{M}(G, S)$. As $f \in \mathcal{M}(G, S)$, for every $T \in \mathcal{F}$ and $v \in V(T)$, $\sum f(N(v)) = k$. Consider a tree $T \in \mathcal{F}$. Let v be a non-leaf vertex of T. Then, for all $w \in \text{children}_T(v)$, $\sum f(N_G(w)) = k \Rightarrow f(v) + \sum f(\text{children}_T(w) \setminus \text{leaves}(T)) + \sum f(\text{children}_T(w) \cap \text{leaves}(T)) + \sum f(N_G(w) \setminus V(T)) = k$. As $f' = f|_{N_G(F)\cup\text{leaves}(F)}$, for all $v \in V(T) \setminus \text{leaves}(T)$ and $w \in \text{children}_T(v)$, $f(v) = k - \sum f(\text{children}_T(w) \setminus \text{leaves}(T)) - \sum f'(\text{children}_T(w) \cap \text{leaves}(T)) - \sum f'(N_G(w) \setminus V(T))$. If w is a leaf node, then $\text{children}_T(w) = \emptyset$. So, we can write $f''' = f|_{N_G(F)\cup V(F)}$ as follows.

(i) For every $v \in N_G(F) \cup \text{leaves}(F)$, $f'''(v) = f'(v)$.
(i) For every $i \in [t]$ and $v \in V(T_i) \setminus \text{leaves}(T_i)$,
  - if v has a leaf child w, then $f'''(v) = k - \sum f'(N_G(w) \setminus V(T))$.
  - else, let w be a child of v, then $f'''(v) = k - \sum f(\text{children}_T(w) \setminus \text{leaves}(T)) - \sum f'(\text{children}_T(w) \cap \text{leaves}(T)) - \sum f'(N_G(w) \setminus V(T))$.

As f'' and $f|_{N_G(F)\cup V(F)}$ have same base case and recursive step, $f'' = f|_{N_G(F)\cup V(F)}$. This also implies that $S'' \subseteq S$. □

The following corollary directly follows from the above lemma.

Corollary 4.4 (*). *Let G be a graph and S be multiset of positive integers. Let F be an induced subgraph of G that is a forest. Then, in time $O(\alpha(S)^{|N_G(F)|+|\text{leaves}(F)|} \cdot |V(G)|)$, we can compute a superset of the set $\mathcal{G}$ of functions from $N_G(F) \cup V(F)$ to S such that for every $f \in \mathcal{M}(G, S)$, there exists a $g \in \mathcal{G}$ such that $g = f|_{N_G(F)\cup V(F)}$.*

Lemma 4.5. *The* Fair-NET *problem is FPT parameterized by $\alpha + \Delta$ for disjoint union of stars.*

Proof. The FPT algorithm is based on ILP. We first give the algorithm and then prove its correctness.

Algorithm: Let (G, k) be an instance of Fair-NET for disjoint union of stars. Thus, $G = K_{1,n_1} + K_{1,n_2} + \ldots + K_{1,n_t}$ for some $t, n_1, n_2, \ldots, n_t \in \mathbb{N}$. Note that $\Delta = \Delta(G) = \max(n_1, n_2, \ldots, n_t)$. Denote $V(K_{1,n_i}) = \{v_i\} \cup B_i$ where v_i is the highest degree vertex in K_{1,n_i} and B_i is the set of all other vertices in K_{1,n_i}, for every $i \in [t]$. Let k be the required S-fairness constant of G. By Observations 3 and 4, G is S-fair if and only if there exists a bijection $f : V(G) \to S$ such that for all $i \in [t]$, $f(v_i) = \sum f(B_i) = k$. So, G is S-fair only if $\alpha_S(k) \geq t$. Let $S' = S \setminus \{s_1 = k, s_2 = k, \ldots, s_t = k\}$ and let $\widehat{\alpha} = \alpha(S')$. Let $\ell_1, \ell_2, \ldots, \ell_{\widehat{\alpha}}$ be the unique labels in S'. Let $\mathcal{D} = \{B_1, B_2, \ldots, B_t\}$. We partition $\mathcal{D}$ into $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_\Delta$ such that for every $i \in [\Delta]$, every set $B \in \mathcal{D}_i$ is of size i. As the number of unique labels are $\widehat{\alpha}$, for every $i \in [\Delta]$, any set in $\mathcal{D}_i$ can have at

[1] S'' may not be a subset of S.

most $\widehat{\alpha}^i$ different label assignments. For every $i \in [\Delta]$, let $\mathcal{L}_i$ be the set of feasible label assignments for $\mathcal{D}_i$, i.e., the label assignments for any set in $\mathcal{D}_i$ for which the sum of the labels is k. For every $i \in [\Delta]$, every label assignment $la \in \mathcal{L}_i$ is a set $\{t^1_{la}, t^2_{la}, \ldots, t^{\widehat{\alpha}}_{la}\}$, where, for every $j \in [\widehat{\alpha}]$, t^j_{la} denoted the number of times label ℓ_j is used in the label assignment la. For every $i \in [\Delta]$ and $la \in \mathcal{L}_i$, we have a variable $n_{i,la}$. For any $i \in [\Delta]$, $la \in \mathcal{L}_i$ and a function $f \in \mathcal{M}(G,S)$, $n_{i,la}$ represents the number of times label assignment la is used in $\mathcal{D}_i$ for f. Then, the algorithm works as follows.

- If $\alpha_S(k) < t$, then return False.
- Solve the following ILP to find $n_{i,la}$, for every $i \in [\Delta]$ and $la \in \mathcal{L}_i$.

$$\forall i \in [\Delta], \sum_{la \in \mathcal{L}_i} n_{i,la} = |\mathcal{D}_i|. \tag{1}$$

$$\forall j \in [\widehat{\alpha}], \sum_{i \in [\Delta]} \sum_{la \in \mathcal{L}_i} n_{i,la} \cdot t^j_{la} = \alpha_{S'}(\ell_j). \tag{2}$$

$$\forall i \in [\Delta], \forall la \in \mathcal{L}_i; n_{i,la} \geq 0. \tag{3}$$

- If the ILP returns a feasible solution, then return True; otherwise, return False.

Correctness: Equation 1 ensures that for every $i \in [\Delta]$, the number of label assignments used in $\mathcal{D}_i$ is equal to the number of sets $\mathcal{D}_i$ has. Equation 2 ensures that for every $j \in [\widehat{\alpha}]$, the number of times label ℓ_j is used is equal to the number of times it appears in S'. For every $i \in [\Delta]$ and for every $B \in \mathcal{D}_i$, all the vertices in B have the same neighborhood which is just a single vertex. Thus by Observation 2, it is sufficient to know the label assignment for B; we can arbitrarily assign labels to the vertices in B once we have decided which labels to use for B. Keeping this interpretation in mind, we now prove the correctness.

Assume first that the algorithm returns True. It means the ILP assigned non-negative integer values for the variables $n_{i,la}$, $i \in [\Delta]$ and $la \in \mathcal{L}_i$ such that Equations 1 and 2 are satisfied. As for every $i \in [\Delta]$, $\mathcal{L}_i$ is the set of feasible label assignments, this implies that we got a label assignment for every $B \in \mathcal{D}$ such that sum of the labels of the vertices in B is k. Thus, G is S-fair.

Conversely, let G be S-fair. Then, by Observations 3 and 4, there exists a bijection $f : V(G) \to S$ such that for all $i \in [t]$, $f(v_i) = \sum f(B_i) = k$. So, $\alpha_S(k) \geq t$. Moreover, every $B \in \mathcal{D}$ has a feasible label assignment so the ILP admits a feasible solution. Thus, the algorithm will return True. As the number of variables $n_{i,la}$ is $O(\Delta \cdot \widehat{\alpha}^{\Delta})$, by Theorem 2.7, the Fair-NET problem is FPT parameterized by $\alpha + \Delta$ for disjoint union of stars. □

Theorem 4.6. Fair-NET *is FPT parameterized by* $\mathtt{fvs} + \alpha + \Delta$.

Proof. Let (G,k) be an instance of Fair-NET. Let k be the required S-fairness constant. Then, we compute a minimum feedback vertex set FVS of G in time $O(5^{|FVS|} \cdot |FVS| \cdot |V(G)|^2)$, using the algorithms given by Chen *et al.* [9]. Let $\mathtt{fvs} = |FVS|$ and $F = V(G) \setminus FVS$. Then by definition of feedback vertex set, $G[F]$ is a forest. Let $\mathcal{F}$ be set of connected components of $G[F]$. So, $\mathcal{F}$ is a collection of trees.

Let T be a tree in $\mathcal{F}$. Let v be a leaf vertex of T such that v is not connected to any vertex in FVS, then $\deg_G(v) = 1$. If $\deg_G(v) = 1$, then by Lemma 4.1, either G is a star and $G = T$ or G is a disjoint union of T and $G[V(G) \setminus V(T)]$. By this argument, for any tree $T \in \mathcal{F}$, either all the leaves of T have at least one neighbor in FVS or none of the leaves are connected to any vertex in FVS. So, we can partition $G = G_1 + G_2$, where G_1 is a connected graph where all the leaves of the forest $G_1[F]$ have at least one neighbor in FVS and G_2 is a disjoint union of stars. Note that, G_2 is an induced subgraph of $G[F]$. By Observation 3, G is S-fair if any only if G_1 is S_1-fair and G_2 is $S \setminus S_1$-fair, for some $S_1 \subseteq S$.

Let L be the set of leaves of $G_1[F]$. As $\Delta(G_1) \leq \Delta, |L| \leq \Delta \cdot \mathtt{fvs}$. By Corollary 4.4, we can compute in time $O(\alpha^{(\Delta+1)\mathtt{fvs}} \cdot |V(G)|)$, a superset $\mathcal{H}$ of the set $\mathcal{G}$ of functions from $V(G_1)$ to S such that for every $f \in \mathcal{M}(G,S)$, there exists a $g \in \mathcal{G}$ such that $g = f|_{V(G_1)}$. We can compute $\mathcal{G}$ from $\mathcal{H}$ by going over every set $h \in \mathcal{H}$ and checking whether G_1 is fair under h. If $\mathcal{G} \neq \emptyset$, then G_1 is $g(V(G_1))$-fair for every function $g \in \mathcal{G}$. Then, for every function $g \in \mathcal{G}$, check whether G_2 is $(S \setminus g(V(G_1)))$-fair, using Lemma 4.5. If for some function $g \in \mathcal{G}$, G_2 is $(S \setminus g(V(G_1)))$-fair, then by Observation 3, G is S-fair. Also, by Corollary 4.4 and Lemma 4.5, the Fair-NET problem is FPT parameterized by $\mathtt{fvs} + \alpha + \Delta$. □

We now prove that Fair-NET is FPT parameterized by $\mathtt{vc} + \alpha$.

Theorem 4.7. Fair-NET *is FPT parameterized by* $\mathtt{vc} + \alpha$.

Proof. The FPT algorithm is based on ILP. We first give the algorithm and then prove its correctness.

Algorithm: Let (G,k) be an instance of Fair-NET. Let k be the required S-fair sum. Let S' be the set of unique labels in S. Note that $|S'| = \alpha(S)$. Let $VC \subseteq V(G)$ be a vertex cover of G of size $\mathtt{vc}$. Let $I = V(G) \setminus VC$. By the definition of vertex cover, I is an independent set of G, i.e., no two vertices in I have an edge between them. We partition I into $I_1, I_2, \ldots, I_m$ such that for every $i \in [m]$, I_i is an inclusion-wise maximal set of vertices in I which have the same neighborhood in G. As I is a independent set, $m \leq 2^{\mathtt{vc}}$. For every $v \in VC$, we define a binary indicator set $\{t^v_1, t^v_2, \ldots, t^v_m\}$, where $t^v_j = 1$ if v is adjacent to the vertices in I_j, otherwise $t^v_j = 0$, for every $j \in [m]$.

By Observation 2, if G is S-fair and if we know the labels of VC under some function $f \in \mathcal{M}(G,S)$, then for every $i \in [m]$, it is sufficient to know the number of times every label is used in I_i under f to get a bijective function $f' : V(G) \to S$ such that $f' \in \mathcal{M}(G,S)$. Keeping this insight in mind, let $n_{i,\ell}$ be a variable whose value (to be computed below) will be interpreted as the number of times label ℓ is used in I_i for some function $f : V(G) \to S$, for every $\ell \in S', i \in [m]$. Then, the algorithm works as follows.

- Construct the set $\mathcal{G}$ containing all possible functions $g : VC \to S$. Note that $|\mathcal{G}| \leq \alpha^{\mathtt{vc}}$.
- For every $g \in \mathcal{G}$ and $\ell \in S'$, let $\alpha_g(\ell)$ denote the number of times ℓ appears in $g(VC)$.
- For every $g \in \mathcal{G}$:
  - Solve the following ILP to find an assignment to the variables $n_{i,\ell}$, for every $i \in [m], \ell \in S'$.

$$\forall v \in VC, \sum_{i \in [m]} t^v_i \sum_{\ell \in S'} n_{i,\ell} \cdot \ell = k \tag{4}$$

$$\forall i \in [m], \sum_{\ell \in S'} n_{i,\ell} = |I_i| \tag{5}$$

$$\forall \ell \in S', \sum_{i \in [m]} n_{i,\ell} = \alpha_S(\ell) - \alpha_g(\ell) \quad (6)$$

$$\forall i \in [m], \forall \ell \in S'; n_{i,\ell} \geq 0. \quad (7)$$

– If the ILP returns a feasible solution, then return True if the following statement holds.

$$\forall i \in [m], \sum_{v \in VC} t_i^v \cdot f(v) = k \quad (8)$$

- Return False.

Correctness: Equation 4 ensures that for every vertex in VC, the neighborhood sum is k. Equation 5 ensures that for every $i \in [m]$, the total number of labels used in I_i is equal to the size of I_i. Equation 6 ensures that for every unique label $\ell \in S'$, the total number of times it is used is equal to the number of times it appear in S. Finally, Equation 8 ensures that for every vertex in IS, the neighborhood sum is k. As for every $i \in [m]$, all the vertices in I_i have the same neighborhood, we only check the sum once per I_i. Keeping this interpretation in mind, we now prove the correctness, i.e. the algorithm returns True if and only if G is S-fair.

Assume first that the algorithm returns True. It means the ILP assigned non-negative integer values to $n_{i,\ell}$, for every $i \in [m]$ and $\ell \in S'$, such that Equations 4, 5 and 6 are satisfied as well as that Equation 8 returned True. As explained above, that implies that for every vertex in G, the neighborhood sum is the same and equals to k. Thus, G is S-fair.

Conversely, let G be S-fair. Then, there exists a bijection $f : V(G) \rightarrow S$ such that for every $v \in V(G), \sum f(N(v)) = k$. As $\mathcal{G}$ is the set of all possible functions from VC to S, $f|_{VC} \in \mathcal{G}$, and hence there exists an iteration where the algorithm examines $g = f|_{VC}$. For $g = f|_{VC}$, the ILP admits a feasible solution. Moreover, Equation 8 then holds. Thus, the algorithm will return True. As $|\mathcal{G}| \leq \alpha^{\mathsf{vc}}$ and the number of variables $n_{i,\ell}$ is at most $2^{\mathsf{vc}} \cdot \alpha$, by Theorem 2.7, Fair-NET is FPT parameterized by $\mathsf{vc} + \alpha$. □

We now prove that the Fair-NET problem is FPT parameterized by fvs for regular graphs.

Theorem 4.8. *The* Fair-NET *problem is FPT parameterized by* fvs *for regular graphs.*

Proof. Let $r \in \mathbb{N}$. Let (G, k) be an instance of Fair-NET for r-regular graphs. Let k be the required S-fairness constant. Then, we compute a minimum feedback vertex set FVS of G in time $O(5^{|FVS|} \cdot |FVS| \cdot |V(G)|^2)$, using the algorithms given by Chen *et al.* [9]. Let $\mathsf{fvs} = |FVS|$ and $F = V(G) \setminus FVS$. Then, $G[F]$ is a forest. We distinguish G into the following three cases.

Case 1 **[**$r = 1$**]:** In this case, G is a collection of edges, i.e., $G = tP_2$ for some $t \in \mathbb{N}$. Then, by Observation 3 and by the definition of S-fair labeling, G is S-fair if and only if $S = \biguplus_{i \in [t]} \{a_i, k - a_i\}$ where for all $i \in [t], a_i \in \{1, 2, \ldots, k\}$. So, we can solve Fair-NET problem in $O(|S| \log S) = O(|V(G)| \log |V(G)|)$ time by sorting S and checking whether S satisfies the above property.

Case 2 **[**$r = 2$**]:** In this case, G is a collection of cycles, i.e. $G = C_{n_1} + C_{n_2} + \ldots + C_{n_t}$ for some $t \in \mathbb{N}$. By the definition of feedback vertex set and the minimality of FVS, every cycle contains exactly one vertex from FVS. So, $t = \mathsf{fvs}$. Also, by Observation 3 and by Lemma 4.2, for every $f \in \mathcal{M}(G, S)$, every cycle is assigned at most 4 distinct labels, by f. So, $\alpha(S) \leq 4\mathsf{fvs}$. As $\Delta(G) = 2$ and $\alpha \leq 4\mathsf{fvs}$, by Theorem 4.6, Fair-NET is FPT parameterized by fvs in this case.

Case 3 **[**$r \geq 3$**]:** As $G[F]$ is a forest, $|E(F)| \leq |V(F)| - 1 \leq |V(G)| - 1$. Also, G is a r-regular graph so $|E(G)| = r \cdot |V(G)|/2$. This implies that, at least $(r/2-1) \cdot |V(G)| + 1$ edges are incident to vertices of FVS. As every vertex of FVS is incident to r edges, $(r/2-1) \cdot |V(G)| + 1 \leq r \cdot \mathsf{fvs}$. Since $r \geq 3$, we get that $|V(G)| = O(fvs)$. As the Fair-NET problem can always be solved in time $O(|V(G)|!)$ using brute-force approach by going over all the permutations of labels, the Fair-NET problem is FPT parameterized by fvs in this case as well. □

Finally, we give a simple lemma proving that the Fair-NET problem is FPT parameterized by $\mathsf{vc} + \Delta$.

Lemma 4.9. Fair-NET *is FPT parameterized by* $\mathsf{vc} + \Delta$.

Proof. Let G be a graph of maximum degree Δ and let $VC \subseteq V(G)$ be a vertex cover of G of size vc. Then, by the definition of vertex cover, it is easy to see that $|V(G)| \leq \mathsf{vc} \cdot (\Delta + 1)$. As the Fair-NET problem can always be solved in time $O(|V(G)|!)$ using brute-force approach by going over all the permutations of labels, the Fair-NET problem is FPT parameterized by $\mathsf{vc} + \Delta$. □

5 CONCLUSION AND FUTURE RESEARCH

In this paper, we initiated a systematic algorithmic study of Fair-NET and presented a comprehensive picture of the parameterized complexity of the problem. We showed NP-hardness results on special graph classes, which implied that the problem is para-NP-hard with respect to several combinations of structural graph parameters. We also showed that the problem is FPT for some combinations of structural graph parameters.

While our work is comprehensive, we stress that it also opens a whole new world of research questions within computational social choice. For illustration, let us mention a few such questions:

(1) Establishing the parameterized complexity of Fair-NET with respect to $\mathsf{tw} + \Delta$.
(2) Establishing the parameterized complexity of Fair-NET with respect to $\mathsf{tw} + \Delta + \alpha$. By employing standard technique over tree decomposition, the problem can be shown FPT with respect to $\mathsf{tw} + \Delta$, when α is a constant. But, the parameterized complexity when α is not a constant is still open.
(3) Establishing the classical complexity of Fair-NET where $S = [n]$, ,where n denotes the number of vertices in the input graph.
(4) Studying the scenario where there is no input infrastructure graph and the objective is to construct one which is S-fair.
(5) Analysis of related labellings such as {0,1}-Fair-NET [6] and vertex-bimagic labeling [4].
(6) Introducing additional fairness notions for non-eliminating tournaments, perhaps by refining/extending/modifying the notion of S-fairness.
(7) Introducing manipulation and bribery to Fair-NET.

ACKNOWLEDGMENTS

The first author is supported in part by the Zuckerman STEM Leadership Program. The second author is supported in part by the Israel Science Foundation grant no. 1176/18, and Binational Science Foundation (BSF) grant no. 2018302.

REFERENCES

[1] S Arumugam, Dalibor Froncek, and N Kamatchi. 2012. Distance magic graphs-a survey. *Journal of the Indonesian Mathematical Society* (2012), 11–26.

[2] S Arumugam, Aloysius Godinho, and Tarkeshwar Singh. 2017. The Distance Magic Index of a Graph. *Discussiones Mathematicae Graph Theory* 38 (08 2017). https://doi.org/10.7151/dmgt.1998

[3] Haris Aziz, Serge Gaspers, Simon Mackenzie, Nicholas Mattei, Paul Stursberg, and Toby Walsh. 2014. Fixing a Balanced Knockout Tournament. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* 552–558. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8352

[4] Baskar J Babujee and S Babitha. 2014. On 1-vertex bimagic vertex labeling. *Tamkang journal of mathematics* 45, 3 (2014), 259–273.

[5] Siddharth Barman, Ganesh Ghalme, Shweta Jain, Pooja Kulkarni, and Shivika Narang. 2019. Fair Division of Indivisible Goods Among Strategic Agents. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019.* 1811–1813. http://dl.acm.org/citation.cfm?id=3331927

[6] S Beena. 2009. On $\sum$ and $\sum'$ labelled graphs. *Discrete Mathematics* 309, 6 (2009), 1783–1787.

[7] Nawal Benabbou, Mithun Chakraborty, Edith Elkind, and Yair Zick. 2019. Fairness Towards Groups of Agents in the Allocation of Indivisible Items. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019.* 95–101. https://doi.org/10.24963/ijcai.2019/14

[8] Aurélie Beynier, Sylvain Bouveret, Michel Lemaître, Nicolas Maudet, Simon Rey, and Parham Shams. 2019. Efficiency, Sequenceability and Deal-Optimality in Fair Division of Indivisible Goods. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019.* 900–908. http://dl.acm.org/citation.cfm?id=3331783

[9] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. 2007. Improved Algorithms for the Feedback Vertex Set Problems. In *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings.* 422–433. https://doi.org/10.1007/978-3-540-73951-7_37

[10] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Springer. https://doi.org/10.1007/978-3-319-21275-3

[11] Reinhard Diestel. 2012. *Graph Theory, 4th Edition.* Graduate texts in mathematics, Vol. 173. Springer.

[12] Rodney G. Downey and Michael R. Fellows. 2013. *Fundamentals of Parameterized Complexity.* Springer. https://doi.org/10.1007/978-1-4471-5559-1

[13] Duncan K Foley. 1967. Resource Allocation and the Public Sector. *Yale Economic Essays 7* (1967), 45–98.

[14] András Frank and Éva Tardos. 1987. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7, 1 (1987), 49–65. https://doi.org/10.1007/BF02579200

[15] Joseph A. Gallian. 2018. A Dynamic Survey of Graph Labeling. *The Electronic Journal of Combinatorics* (2018).

[16] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman.

[17] Mohammad Ghodsi, Mohammad Taghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami. 2018. Fair Allocation of Indivisible Goods: Improvements and Generalizations. In *Proceedings of the 2018 ACM Conference on Economics and Computation, Ithaca, NY, USA, June 18-22, 2018.* 539–556. https://doi.org/10.1145/3219166.3219238

[18] Aloysius Godinho, Tarkeshwar Singh, and S. Arumugam. 2015. On S-Magic Graphs. *Electronic Notes in Discrete Mathematics* 48 (2015), 267–273. https://doi.org/10.1016/j.endm.2015.05.040

[19] Marc Goerigk, Richard Hoshino, Ken-ichi Kawarabayashi, and Stephan Westphal. 2014. Solving the Traveling Tournament Problem by Packing Three-Vertex Paths. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.* 2271–2277. http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8345

[20] Sushmita Gupta, Sanjukta Roy, Saket Saurabh, and Meirav Zehavi. 2018. When Rigging a Tournament, Let Greediness Blind You. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.* 275–281. https://doi.org/10.24963/ijcai.2018/38

[21] Siddharth Gupta and Meirav Zehavi. 2021. Multivariate Analysis of Scheduling Fair Competitions. arXiv:2102.03857 [cs.DS]

[22] Richard Hoshino and Ken-ichi Kawarabayashi. 2011. The Inter-League Extension of the Traveling Tournament Problem and its Application to Sports Scheduling. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011.* http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/view/3459

[23] Richard Hoshino and Ken-ichi Kawarabayashi. 2012. The Linear Distance Traveling Tournament Problem. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada.* http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4862

[24] FK Hwang. 1982. New concepts in seeding knockout tournaments. *The American Mathematical Monthly* 89, 4 (1982), 235–239.

[25] Hendrik W. Lenstra Jr. 1983. Integer Programming with a Fixed Number of Variables. *Math. Oper. Res.* 8, 4 (1983), 538–548. https://doi.org/10.1287/moor.8.4.538

[26] V. Vilfred Kamalappan. 2006. SIGMA PARTITION AND SIGMA LABELED GRAPHS. *J. of Decision and Math. Sci.* 10 (01 2006), 1–12.

[27] Ravi Kannan. 1987. Minkowski's Convex Body Theorem and Integer Programming. *Math. Oper. Res.* 12, 3 (1987), 415–440. https://doi.org/10.1287/moor.12.3.415

[28] Michael P. Kim, Warut Suksompong, and Virginia Vassilevska Williams. 2016. Who Can Win a Single-Elimination Tournament?. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.* 516–522. http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12194

[29] Michael P. Kim and Virginia Vassilevska Williams. 2015. Fixing Tournaments for Kings, Chokers, and More. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015.* 561–567. http://ijcai.org/Abstract/15/085

[30] Christine Konicki and Virginia Vassilevska Williams. 2019. Bribery in Balanced Knockout Tournaments. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019.* 2066–2068. http://dl.acm.org/citation.cfm?id=3332012

[31] Rafael A. Melo, Sebastián Urrutia, and Celso C. Ribeiro. 2009. The traveling tournament problem with predefined venues. *J. Scheduling* 12, 6 (2009), 607–622. https://doi.org/10.1007/s10951-008-0097-1

[32] Mirka Miller, Chris Rodger, and Rinovia Simanjuntak. 2003. Distance magic labelings of graphs. *Australasian J. Combinatorics* 28 (2003), 305. http://ajc.maths.uq.edu.au/pdf/28/ajc_v28_p305.pdf

[33] Allen O'Neal and Peter J. Slater. 2013. Uniqueness of Vertex Magic Constants. *SIAM J. Discrete Math.* 27, 2 (2013), 708–716. https://doi.org/10.1137/110834421

[34] Stefan Porschen, Tatjana Schmidt, Ewald Speckenmeyer, and Andreas Wotzlaw. 2014. XSAT and NAE-SAT of linear CNF classes. *Discrete Applied Mathematics* 167 (2014), 1–14. https://doi.org/10.1016/j.dam.2013.10.030

[35] M. S. Ramanujan and Stefan Szeider. 2017. Rigging Nearly Acyclic Tournaments Is Fixed-Parameter Tractable. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.* 3929–3935. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14706

[36] Celso C. Ribeiro and Sebastián Urrutia. 2006. Scheduling the Brazilian Soccer Tournament with Fairness and Broadcast Objectives. In *Practice and Theory of Automated Timetabling VI, 6th International Conference, PATAT 2006, Brno, Czech Republic, August 30 - September 1, 2006, Revised Selected Papers.* 147–157. https://doi.org/10.1007/978-3-540-77345-0_10

[37] Rachel Rupnow. 2014. A survey of distance magic graphs. *Master's report, Michigan Technological University* (2014). https://digitalcommons.mtu.edu/etds/829

[38] Philip A. Scarf, Muhammad Mat Yusof, and Mark Bilbao. 2009. A numerical study of designs for sporting contests. *European Journal of Operational Research* 198, 1 (2009), 190–198. https://doi.org/10.1016/j.ejor.2008.07.029

[39] Allen J. Schwenk. 2000. What is the Correct Way to Seed a Knockout Tournament? *The American Mathematical Monthly* 107, 2 (2000), 140–150. http://www.jstor.org/stable/2589435

[40] Peter J. Slater. 2016. *It Is All Labeling.* Springer International Publishing, Cham, 231–252. https://doi.org/10.1007/978-3-319-31940-7_14

[41] Isabelle Stanton and Virginia Vassilevska Williams. 2011. Manipulating Stochastically Generated Single-Elimination Tournaments for Nearly All Players. In *Internet and Network Economics - 7th International Workshop, WINE 2011, Singapore, December 11-14, 2011. Proceedings.* 326–337. https://doi.org/10.1007/978-3-642-25510-6_28

[42] Isabelle Stanton and Virginia Vassilevska Williams. 2011. Rigging Tournament Brackets for Weaker Players. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011.* 357–364. https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-069

[43] KA Sugeng, D Fronček, M Miller, J Ryan, and J Walker. 2009. On distance magic labeling of graphs. *Journal of Combinatorial Mathematics and Combinatorial Computing* 71 (2009), 39–48.

[44] David C. Uthus, Patricia J. Riddle, and Hans W. Guesgen. 2012. Solving the traveling tournament problem with iterative-deepening A_*. *J. Scheduling* 15, 5 (2012), 601–614. https://doi.org/10.1007/s10951-011-0237-x

[45] Pim van 't Hof, Gerhard F. Post, and Dirk Briskorn. 2010. Constructing fair round robin tournaments with a minimum number of breaks. *Oper. Res. Lett.* 38, 6 (2010), 592–596. https://doi.org/10.1016/j.orl.2010.08.008

[46] Thuc Vu, Alon Altman, and Yoav Shoham. 2009. On the complexity of schedule control problems for knockout tournaments. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary, May 10-15, 2009, Volume 1.* 225–232. https://dl.acm.org/citation.cfm?id=1558044

[47] Thuc Vu and Yoav Shoham. 2010. Optimal seeding in knockout tournaments. In *9th International Conference on Autonomous Agents and Multiagent Systems*

(AAMAS 2010), Toronto, Canada, May 10-14, 2010, Volume 1-3. 1579–1580. https://dl.acm.org/citation.cfm?id=1838490

[48] Thuc Vu and Yoav Shoham. 2011. Fair Seeding in Knockout Tournaments. *ACM TIST* 3, 1 (2011), 9:1–9:17. https://doi.org/10.1145/2036264.2036273

[49] Wikipedia contributors. 2020. Magic square — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Magic_square&oldid=936094943

[50] Virginia Vassilevska Williams. 2010. Fixing a Tournament. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1726

[51] Mingyu Xiao and Shaowei Kou. 2016. An Improved Approximation Algorithm for the Traveling Tournament Problem with Maximum Trip Length Two. In *41st International Symposium on Mathematical Foundations of Computer Science, MFCS 2016, August 22-26, 2016 - Kraków, Poland*. 89:1–89:14. https://doi.org/10.4230/LIPIcs.MFCS.2016.89

[52] Lishun Zeng and Shinji Mizuno. 2013. Constructing fair single round robin tournaments regarding strength groups with a minimum number of breaks. *Oper. Res. Lett.* 41, 5 (2013), 506–510. https://doi.org/10.1016/j.orl.2013.06.007