Constructing extremal triangle-free graphs using integer programming

Ali Erdem Banak, Tınaz Ekim, Z. Caner Taşkın
Boğaziçi University, Department of Industrial Engineering

August 20, 2023

Abstract

The maximum number of edges in a graph with matching number m and maximum degree d has been determined in [1] and [2], where some extremal graphs have also been provided. Then, a new question has emerged: how the maximum edge count is affected by forbidding some subgraphs occurring in these extremal graphs? In [3], the problem is solved in triangle-free graphs for $d \geq m$, and for $d < m$ with either $Z(d) \leq m < 2d$ or $d \leq 6$, where $Z(d)$ is approximately $5d/4$. The authors derived structural properties of triangle-free extremal graphs, which allows us to focus on constructing small extremal components to form an extremal graph. Based on these findings, in this paper, we develop an integer programming formulation for constructing extremal graphs. Since our formulation is highly symmetric, we use our own implementation of Orbital Branching to reduce symmetry. We also implement our integer programming formulation so that the feasible region is restricted iteratively. Using a combination of the two approaches, we expand the solution into $d \leq 10$ instead of $d \leq 6$ for $m > d$. Our results endorse the formula for the number of edges in all extremal triangle-free graphs conjectured in [3].

Keywords: Extremal graphs; factor-critical; integer programming; orbital branching.

1 Introduction

Extremal graph theory studies how big or small a graph parameter (usually the number of edges or vertices) can be under some local constraints. A graph that is an optimal solution of such a maximization or minimization problem is called an *extremal graph*. Turan's graphs are one of the best known examples of extremal graph theory results [4]. The extremal problem of maximizing the number of edges in a graph while limiting the size of a maximum matching and the maximum degree has been first posed by Erdös and Rado [5] in a more general context. This question has been first answered by Chvátal and Hanson in [1] using some optimization techniques and Berge's matching formula. Later, Balachandran and Khare [2] provided a constructive proof for the same problem. The authors describe some extremal graphs having three types of components, namely star graphs, complete graphs, and almost complete graphs (which contain C_4's, that is induced cycles of length 4). This construction paves the way for new variants of the problem; how the maximum number of edges will be affected if we exclude one type of component described in [2]? Accordingly, one can consider the same extremal problem for graphs (independently) restricted to be claw-free, which forbids the smallest star graph; triangle-free, which forbids the smallest complete graph; or C_4-free, which forbids almost-complete graphs.

Dibek, Ekim, and Heggernes studied claw-free graphs and settled the discussion by showing the cases where the maximum number of edges achieves the maximum number for general graphs and where it is strictly less (than for the general case) [6]. Blair, Heggernes, Lima, and Lokshtanov worked on chordal graphs, which exclude C_4's and therefore almost-complete graphs. They show that the maximum edge count in chordal graphs is the same as the general graphs [7]. Similarly, Maland [8] answers the question on bipartite graphs, split graphs, disjoint union of split graphs, and unit interval graphs.

Triangle-free graphs have been considered from the same perspective by Ahanjideh, Ekim, and Yıldız in [3]. In their study, the authors provided some partial answers to the extremal problem, which

is thus not completely solved yet. The authors constructed extremal graphs for all $d \geq m$. For the cases with $d < m$, they solved the problem for either $d \leq 6$ or $Z(d) \leq m < 2d$ where $Z(d)$ is approximately $5d/4$. Consequently, the cases with $d \geq 7$ for either $m \geq 2d$ or $d < m < Z(d)$ are left open in [3]. Nevertheless, the authors derived some structural results that shed light on these open cases: For all natural numbers $d \geq 2$ and m, there exists an extremal graph whose components are either star graphs or factor-critical triangle-free extremal graphs with maximum degree d and matching number between d and $Z(d)$. This result implies that to construct extremal graphs for $2d \leq m$, it is sufficient to find extremal components, that is, extremal graphs with matching number between d and $Z(d)$ and decide how many of each one of those components and how many star graphs should be taken. This problem is then expressed as a Knapsack Formulation where the total volume is m, individual volumes are the matching numbers of each extremal component, and utilities are maximum edge counts in each extremal component.

Following the above observation, in this paper, we focus on the computation of maximum edge counts in extremal components and propose integer programming approaches to find them. To the best of our knowledge, the use of integer programming formulations in the construction of extremal graphs is rather new. A few exceptions are the recent paper on Ramsey numbers to improve the known lower bounds [9] and the construction of (small) extremal chemical graphs with given number of vertices and/or degrees optimizing some invariants [10], [11]. In Section 2, we introduce the notation and mention the known results for general graphs and triangle-free graphs. We also provide the knapsack formulation in [3] to compute the max edge counts for $d \geq 7$ and $m \geq 2d$. We state Conjecture 2.1 and Conjecture 2.2, giving the formula for the max edge counts of extremal components and all triangle-free extremal graphs respectively. In particular, we exhibit the results in [3], which allows us to restrict our study to the construction of extremal factor-critical triangle-free components with $d > 6$ and $d < m < Z(d)$. In Section 3, we explain our methodology to construct the extremal components via four different exact methods. Our Basic Formulation exploits the structural information provided in [3] to bound the matching number without explicitly including it in the formulation. Noting that the Basic Formulation is highly symmetric, we propose various methods to reduce symmetry. In particular, we utilize Orbital Branching [12], which identifies equivalent variables and sets the value of more than one of them in a branch to reduce the symmetry within the branch-and-cut algorithm. Next, Iterative Method solves the Basic Formulation by considering only a portion of the feasible region at each iteration. In this approach, we set all the degrees of vertices to d and check if the problem is feasible. If not, we iteratively check the existence of graphs with the next possible upper bound for the edge count. Finally, we combine both methods, applying Orbital Branching within the Iterative Method. In Section 4, we discuss our implementation, where we used CPLEX 20.1.0 for integer programming and nauty for calculating the orbits.

In Section 5, we provide our computational results. We show that the Orbital Branching and the Iterative Method are both better than the Basic Formulation using the default branching strategy; moreover, their combination gives us the best results. Our method allows us to construct all extremal components for $d = 7, 8, 9, 10$ and $d < m < Z(d)$. Subsequently, we use the maximum edge counts for these extremal components as the parameters of the Knapsack Formulation proposed in [3]. We solve the Knapsack Formulation for $d = 7, 8, 9, 10$ within 0.2 seconds even for $m = 2000$. Therefore, the edge counts of edge-extremal graphs for all possible m values are found for $d = 7, 8, 9, 10$. We could also construct some (but not all) extremal components for $d = 11, 12$ and 13 with $d < m < Z(d)$. Our findings support the conjectures suggested by Ahanjideh, Ekim, and Yıldız in [3] and explained in Section 2. In particular, we now have a stronger evidence that the formula provided in [3] (see Theorem 2.2 and Conjecture 2.2) gives the maximum number of edges in a triangle-free graph with maximum degree at most d and matching number at most m for all natural numbers $d \geq 2$ and m. Yet, a formal proof remains as a future research.

2 Notation and Preliminaries

In this paper, undirected graphs are represented with $G = (V(G), E(G))$ where $V(G)$ is the set of vertices and $E(G)$ is the set of edges of the graph G. The *degree* of a vertex v, denoted by $d(v)$, is the number of vertices adjacent to v. The *maximum degree* of a graph G is the maximum degree of a vertex in G, denoted by $\Delta(G)$. A *matching* of a graph G is defined as a set of edges with no common vertices. The size of a maximum matching of a graph G is denoted by $\nu(G)$.

Maximum edge count for a general graph with maximum degree at most d and matching number at most m is shown with $f_{GEN}(d,m)$. It can be observed that if we do not limit either $\Delta(G)$ or $\nu(G)$, a graph can have an unlimited number of edges; such examples are a central vertex with an unbounded number of neighbors, and an unbounded number of independent edges respectively. Therefore, we are interested in the number of edges in a graph where $\nu(G)$ is bounded by m and $\Delta(G)$ is bounded by d.

Triangle-free graphs are defined as graphs with no cycle on three vertices. In other words, the size of the largest complete subgraph in a (non-empty) triangle-free graph is two. Let Δ denote the class of all triangle-free graphs. Then, the maximum number of edges in a triangle-free graph with maximum degree at most d and maximum matching size at most m is denoted by $f_{\Delta}(d,m)$.

An ℓ-star $K_{1,\ell}$ is a bipartite graph with one (central) vertex on one side, which is adjacent to ℓ vertices on the other side. If all the vertices of a graph have the same degree d, we call it *d-regular*. If only one vertex has degree $d-1$ and the rest of the vertices have degree d, we call it *almost d-regular*. A graph is called *factor-critical* if removing any vertex from G leaves a graph that admits a perfect matching, that is, a matching saturating all vertices.

The following theorem in [2] gives the formula for $f_{GEN}(d,m)$ for all d and m along with a construction for extremal graphs.

Theorem 2.1. *[2] For general graphs with $\Delta(G) \leq d$ and $\nu(G) \leq m$, the maximum number of edges in an extremal graph is*

$$f_{GEN}(d,m) = dm + \lfloor d/2 \rfloor \lfloor m/\lceil d/2 \rceil \rfloor.$$

Moreover, an extremal graph with $f_{GEN}(d,m)$ edges can be obtained by taking the disjoint union of r copies of d-star and q copies of

$$\begin{cases} K_{d+1} & \text{if } d{+}1 \text{ is odd} \\ K'_{d+1} & \text{if } d{+}1 \text{ is even,} \end{cases}$$

where q is the largest integer such that $m = q\lceil d/2 \rceil + r$ and $r \geq 0$ and where K'_{d+1} is the graph obtained by removing a perfect matching from the complete graph K_{d+1} on $d+1$ vertices, adding a new vertex v, and making v adjacent to d of the other vertices.

The same extremal problem when restricted to triangle-free graphs has been addressed by Ahanjideh, Ekim, and Yıldız in [3]. The authors settle all the cases for $d \geq m$, and for $d < m$ with either $d \leq 6$ or $Z(d) \leq m < 2d$ where $Z(d)$ is defined as follows.

Definition 2.1. *[3] For any $d \geq 2$, let $Z(d)$ be the smallest natural number n such that there exists a d-regular (if d is even) or almost d-regular (if d is odd) triangle-free and factor-critical graph G with $\nu(G) = n$.*

Since $Z(d)$ plays a crucial role in the description of triangle-free extremal graphs, the authors first describe a graph construction that proves the existence of $Z(d)$, then investigate further the value of $Z(d)$.

Lemma 2.1. *[3] We have $Z(d) = d$ for $d \in \{2,3\}$ and $Z(d) = d+1$ for $d \in \{4,5\}$.*

Lemma 2.2. *[3] For $d \geq 2$, if d is even then we have $Z(d) = \lfloor 5d/4 \rfloor$; if d is odd then we have $\lfloor 5(d-1)/4 \rfloor \leq Z(d) \leq \lceil 5(d+1)/4 \rceil$.*

The main result in [3] provides the following formula for $f_{\Delta}(d,m)$ in all solved cases along with a description of extremal graphs, which we omit here for the sake of brevity.

Theorem 2.2. *[3] Let d and m be natural numbers with $d \geq 2$, and let k and r be non-negative integers such that $m = kZ(d) + r$ with $0 \leq r < Z(d)$. Then, for all cases with $d \geq m$, and for the cases $d < m$ with either $d \leq 6$ or $Z(d) \leq m < 2d$, we have*

$$f_{\Delta}(d,m) = \begin{cases} dm + k\lfloor d/2 \rfloor, & \text{if } r < d \\ dm + k\lfloor d/2 \rfloor + r - d + 1, & \text{if } r \geq d. \end{cases}$$

It follows from Theorem 2.2 that the remaining open cases are $d \geq 7$ with either $m \geq 2d$ or $d < m < Z(d)$. Apart from the formula for $f_\Delta(d, m)$, an important contribution in [3] is the following result, which shows that there is a triangle-free extremal graph with a special structure. This structural property expressed as a combination of several results (namely Corollary 2.2 and Lemma 4.4) in [3], is crucial in building our integer programming formulations.

Lemma 2.3. *[3] Let d and m be natural numbers with $d \geq 2$, and let G be an edge-extremal graph with maximum number of connected components isomorphic to a d-star. Then, for every connected component H of G, one of the following is true:*
(i) H is a d-star.
(ii) H is factor-critical with $|E(H)| = f_\Delta(d, \nu(H))$ and $|V(H)| = 2\nu(H) + 1$ where $d \leq \nu(H) \leq Z(d)$.

Now, consider the open case for $m \geq 2d$ and $d \geq 7$. Lemma 2.3 states that there is a triangle-free edge-extremal graph whose components which are not d-stars are edge-extremal factor-critical triangle-free graphs with matching number between d and $Z(d)$. Let us call the latter *extremal components* throughout the paper. This information is used in [3] to formulate the open cases with $m \geq 2d$ and $d \geq 7$ as a knapsack problem where the utility parameters are $f_\Delta(d, i)$ for $d \leq i \leq Z(d)$, which are yet to be calculated. Let x_i be the number of extremal components of G with matching number i. Then an extremal graph for d and m can be obtained as follows. For $d \leq i \leq Z(d)$, take x_i many extremal components with matching number i; their contribution to the total number of edges $f_\Delta(d, m)$ is the second summation on the left-hand side of Equation (1), and they contribute $\sum_{i=d}^{Z(d)} ix_i$ to the matching number of the graph. Complete the matching number to m using d-stars, each of which adding d to $f_\Delta(d, m)$; this is expressed by the first term on the left-hand side of Equation (1). Then we have the following:

$$f_\Delta(d, m) = d(m - \sum_{i=d}^{Z(d)} ix_i) + \sum_{i=d}^{Z(d)} f_\Delta(d, i)x_i = dm + \sum_{i=d}^{Z(d)} (f_\Delta(d, i) - di)x_i. \tag{1}$$

Based on these observations, Ahanjideh, Ekim, and Yıldız propose the following bounded knapsack formulation where the utility of item i is $(f_\Delta(d, i) - di)$ and its volume is i [3].

$$\text{(Knapsack Formulation)} \quad \max dm + \sum_{i=d}^{Z(d)} (f_\Delta(d, i) - di)x_i \tag{2}$$

$$\text{s.t.} \quad \sum_{i=d}^{Z(d)} ix_i \leq m \tag{3}$$

$$x_i \geq 0, \quad x_i \in \mathbb{Z} \tag{4}$$

With this formulation, if the edge counts of the extremal components are known for a fixed d, then edge extremal graphs for all m can be found. Therefore, in this paper, our effort is focused on finding the extremal components, that is the edge-extremal graphs H for $d < m$ with $d > 6$ and $d < \nu(H) < Z(d)$. Note that the extremal components with matching number d and $Z(d)$ follow from Theorem 2.2; we have $f_\Delta(d, d) = d^2 + 1$ and $f_\Delta(d, Z(d)) = dZ(d) + \lfloor \frac{d}{2} \rfloor$. However, $Z(d)$ is not known when d is odd.

Although the authors in [3] leave the development of further methods to solve this knapsack formulation, they still provide some insights about its solution. In particular, they conjecture that its unknown parameters, namely $f_\Delta(d, i)$ for $7 \leq d < i < Z(d)$ follow the formula given in Theorem 2.2, which simplifies as follows in this case:

Conjecture 2.1. *[3] For $7 \leq d < i < Z(d)$, we have $f_\Delta(d, i) = di + i - d + 1$.*

The authors also show that if Conjecture 2.1 holds then the Knapsack Formulation admits a special optimal solution having as many extremal components as possible with matching number $Z(d)$, and at most one extremal component with smaller matching number.

Proposition 2.1. *[3] If Conjecture 2.1 is true, then for $7 \leq d < m < Z(d)$, the Knapsack Formulation admits an optimal solution with $\sum_{i=d}^{Z(d)-1} x_i \leq 1$. In other words, $x_{Z(d)}$ is maximized and there is at most one other x_i which is 1 (all the rest being zero).*

Based on the observation that the description of the extremal graphs in Proposition 2.1 is similar to those provided in Theorem 2.2 (which we omitted in our paper), the authors also conjecture that the formula in Theorem 2.2 holds in general (including the open cases).

Conjecture 2.2. *[3] For all natural numbers $d \geq 2$ and m, let $m = kZ(d) + r$. Then we have*

$$f_\Delta(d,m) = \begin{cases} dm + k\lfloor d/2 \rfloor, & \text{if } r < d \\ dm + k\lfloor d/2 \rfloor + r - d + 1, & \text{if } r \geq d. \end{cases}$$

Next, we develop integer programming formulations to construct extremal components. Our method allows us to construct all extremal components for $d = 7, 8, 9, 10$ and consequently to solve the Knapsack Formulation for these d values (and any m). All our findings support Conjecture 2.1, Proposition 2.1, and Conjecture 2.2; and therefore strengthen them. As a byproduct, we also obtain some new values for $Z(d)$, namely, $Z(7) = 9, Z(9) = 13, Z(11) = 15$, and $Z(13) = 17$.

3 Methodology

We formulate the construction of an edge extremal triangle-free graph with a matching number at most m and degree at most d as an integer programming problem. Let V be the vertex set and let x_{ij} be a binary variable defined for $i \neq j$ that takes on value 1 if there is an edge between $i, j \in V$ and 0 otherwise. We work with undirected graphs; therefore, we only use one of x_{ij} and x_{ji} variables. For notational simplicity in the models, ij in x_{ij} should be considered an unordered set while the edges are only defined for $i > j$.

$$\text{(Basic Formulation)} \quad \max \sum_{i,j \in V} x_{ij} \tag{5}$$

$$\text{s.t.} \quad x_{ij} + x_{jk} + x_{ik} \leq 2 \quad \forall i,j,k \in V \tag{6}$$

$$\sum_{j \in V} x_{ij} \leq d \quad \forall i \in V \tag{7}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{8}$$

The objective function (5) maximizes the edge count. Constraint (6) ensures that the resulting graph is triangle-free and Constraint (7) limits the maximum degree. We do not bound the matching number explicitly; instead, we make use of Lemma 2.3, which guarantees the existence of an edge-extremal triangle-free graph that is factor-critical and fix $|V| = 2m + 1$ accordingly. Clearly, this bounds the matching number with m. Since there is a triangle-free graph with $2m + 1$ vertices and degree bounded by d for every d and m, an optimal solution of the Basic Formulation describes an extremal triangle-free graph having $f_\Delta(d, m)$ edges. The graph resulting from the Basic Formulation might or might not be factor-critical; however, Lemma 2.3 guarantees that it has $f_\Delta(d, m)$ edges because there is at least one extremal component which is factor-critical.

The Basic Formulation is highly symmetric since all x_{ij}'s are interchangeable. To overcome this problem, we propose two approaches and combine them as a third approach. Our first approach is Orbital Branching, which modifies the branching decisions during the solution of Basic Formulation via the branch-and-cut algorithm. Orbital Branching helps us eliminate symmetric solutions in the branch-and-cut tree. Our second approach is solving the problem with minor modifications iteratively, which we call the Iterative Method. In the first iteration, all the vertex degrees are set to d. If there is no feasible solution, all the degrees are set to d but one, which is constrained to be less than d, reducing the upper bound of the solution. As long as there is no feasible solution, the degree of one more vertex is constrained to be less than d at every iteration. This approach basically restricts the feasible region of the problem.

3.1 Orbital Branching

We use Orbital Branching to modify the branching decisions while solving the Basic Formulation. The idea is to branch simultaneously on multiple variables on the same symmetry group instead of branching on a single variable at every node of the branch-and-cut tree.

Before explaining the symmetry groups and equivalency of variables in formal terms, let us first investigate the structure of our Basic Formulation. All x_{ij}'s are interchangeable since nothing distinguishes one edge from others in the formulation. Assume that we create a branch for each variable $x_{12} = 1$, $x_{13} = 1$, and so on, as well as a branch where all variables are 0 to cover the feasible region. This is a legitimate branching strategy since the entire feasible region is covered, even if there are intersections between branches. However, we can observe that each child node with $x_{1j} = 1, j \in V \setminus \{1\}$ yields the same subproblem. Therefore, instead of branching on $|V|$ nodes, we can branch on only two nodes by selecting a representative node, say $x_{12} = 1$, and creating another branch with $x_{12} = x_{13} = ... = 0$. This is the main idea behind Orbital Branching. In the root node, it is easy to see the equivalent variables; but after the first branching, it gets more complicated. Now, x_{13} is not equivalent to x_{34} since they are creating different subproblems due to the already fixed variable $x_{12} = 1$ at that stage. Therefore, we need a formal definition of symmetry groups to identify them. To this end, let us consider the following problem:

$$\max_{x \in \{0,1\}^n} \{c^T x | Ax \leq b\}. \tag{9}$$

A symmetry group of matrix A is defined as the set of permutations which leaves A invariant [13]. This means that there exists a permutation of variables (columns) and constraints (rows) such that applying them consecutively on A creates the same matrix A [12]. If the permutation of columns also creates the same c and the permutation of rows creates the same b; the new problem is equivalent to the first one. Symmetry groups of an IP can be determined via graph automorphism. The IP is converted into a bipartite graph $B(V, M, E)$, where V represents variables and M represents constraints. There is an edge (v_i, m_j) if only if $A_{ij} \neq 0$, that is, variable i occurs in constraint j with a non-zero coefficient. To account for the differences in c, b, and A values, a color code is used to show interchangeable vertices in B [14]. In general, for every cost coefficient c_i value, a different color is used for vertex $v_i \in V$. Similarly, vertex $m_j \in M$ is colored according to b_j and the inequality type of the constraint. In this setting, changing the order of columns is equivalent to changing the labels of vertices in V, and changing the order of rows is equivalent to changing the labels of vertices in M. Note that we should prevent interchanging a vertex from V with a vertex from M, so they are also assigned different colors. Then, the symmetry groups of the IP and the automorphism groups of the graph $B(V, M, E)$ are the same. If there exists an automorphism assigning x_i to $x_{i'}$, they are deemed equivalent and part of the same orbit. Orbits consist of equivalent variables and they are used for branching decisions in Orbital Branching.

Calculating automorphism groups divides the vertices into orbits, which are set of equivalent vertices. There are two sets of orbits, variable orbits, and constraint orbits. For our Orbital Branching approach, we use the variable orbits and prioritize the orbit having the maximum number of elements to branch on.

In Figure 2 an illustrative example of Orbital Branching is provided for the model below.

$$\begin{aligned}
\max \quad & x + y + z + t && (10)\\
\text{s.t.} \quad & x + y \geq 1 && (11)\\
& x + y \geq 1 && (12)\\
& x + y \geq 1 && (13)\\
& x, y, z, t \in \{0, 1\} && (14)
\end{aligned}$$

In the graph representation of the model (10)-(14) in Figure 1, the vertices x, y, z, t represent the variables and vertices c_1, c_2, c_3 represent the constraints. Let the graph on the left side of Figure 1 be the original bipartite graph. Then, we can see that interchanging vertex labels y, z, and c_1, c_2 create the same incidence matrix. This is equivalent to saying that the graph on the right is obtained by relabeling the vertices of the graph on the left differently, thus the two graphs in Figure 1 are isomorphic. This procedure is represented with $(y, z)(c_1, c_2)$ and it is an automorphism group of the bipartite graph.

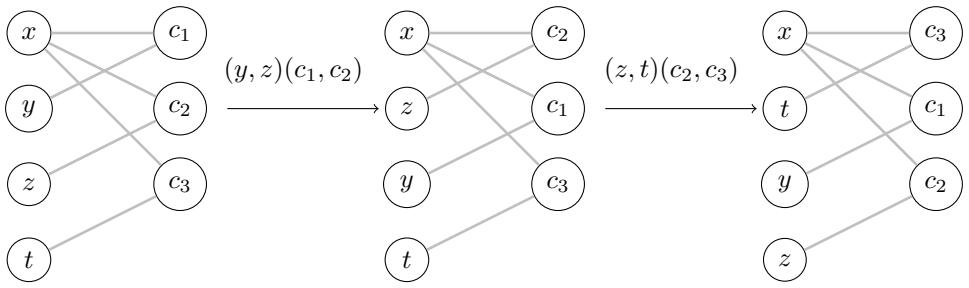

Figure 1: An automorphism of the bipartite graph corresponding to the model (10)-(14)

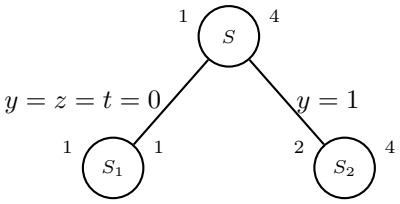

Figure 2: Branch-and-cut tree using Orbital Branching

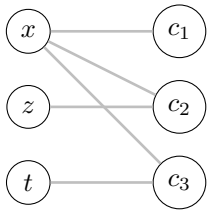

Figure 3: Bipartite graph for S_2

Another automorphism group is $(z,t)(c_2,c_3)$. The application of automorphism groups consecutively yields also an automorphism group. Therefore, we can see that y,z,t are interchangeable and share the same orbit. Consequently, Orbital Branching suggests the branching shown in Figure 2, where the upper and lower bounds for each node are written over the right and left of each node respectively. In each node of at depth 1 of the branch-and-cut tree, an optimal solution is found and the orbits are recalculated based on the new graph. The bipartite graph representing the subproblem S_2 is given in Figure 3 as an example. Let F_0^a and F_1^a denote the variables set to 0 and 1 in node a of a branch-and-cut tree respectively. Then the Orbital Branching algorithm can be summarized as in Algorithm 1:

Algorithm 1 Orbital Branching

Step 1: Given a branch-and-cut node $a = (F_0^a, F_1^a)$, calculate the set of orbits O_a at node a
Step 2: Pick an arbitrary orbit $O \in O_a$
Step 3: Pick an arbitrary variable $v_i \in O$, return new nodes $l = (F_0^a \cup \{O\}, F_1^a)$, $r = (F_0^a, F_1^a \cup \{v_i\})$

We note that all c values are 1 in our Basic Formulation; therefore we do not need to color variable vertices.

3.2 Iterative Method

Since the maximum degree of a vertex and the number of vertices are limited, the *precomputed upper bound* of the objective function value of the Basic Formulation is $\lfloor (2m+1) * d/2) \rfloor$. This bound can only be achieved if all vertices have the maximum degree d. If exactly one of them has degree $d-1$ and all the others have degree d, then the upper bound may decrease since the expression value before rounding down decreases by $1/2$. It is possible to use this information to decrease iteratively the upper bound of the Basic Formulation. For instance, let $d = 8$ and $m = 9$; thus there are $2 * 9 + 1 = 19$ vertices. We have $(8 * 19)/2 = 76$ as an upper bound. We can set all degrees to 8 and check if there is a feasible solution. If there is a feasible solution, it is an optimal solution for the Basic Formulation. If the problem is infeasible, we know that an upper bound of the problem is 75 and at least one of the vertices has a degree at most $d-1$.

Our Iterative Method starts by fixing all the degrees to the degree bound d, solving the resulting integer program, and decreasing the upper bound one by one. Each iteration works on a portion of the original feasible region and decreases the upper bound iteratively. The idea is to obtain an optimal solution to the Basic Formulation by solving the formulation in a restricted feasible region. We can

consider different formulations of this approach, enumerating all possible degree combinations for each objective value or simply decreasing the upper bound by setting the degrees of some vertices to be less than d while the rest are equal to d. We adopt the latter formulation since enumerating all possible degrees would imply an exponential number of iterations.

We distinguish two sets of vertices; let V_d be the set of vertices of degree equal to d and V_{d-1} be the set of vertices of degree at most $d-1$. The formulation can be given as:

$$\text{(Iterative Formulation)} \quad \max \sum_{i,j \in V} x_{ij} \tag{15}$$

$$\text{s.t.} \quad x_{ij} + x_{jk} + x_{ik} \leq 2 \quad \forall i,j,k \in V \tag{16}$$

$$\sum_{j \in V} x_{ij} = d \quad \forall i \in V_d, \tag{17}$$

$$\sum_{j \in V} x_{ij} \leq d-1 \quad \forall i \in V_{d-1}, \tag{18}$$

$$x_{ij} \in \{0,1\} \quad \forall i,j \in V \tag{19}$$

It should be noted that an optimal solution to the Iterative Formulation with $|V_{d-1}| = 1$ might not be an optimal solution for the Basic Formulation. Indeed, a graph where all vertices have degree d but one which has degree $d-4$ has fewer edges than a graph with $|V(G)|-2$ vertices of degree d, and two vertices of degree $d-1$. Consequently, the optimal solution of the Iterative Formulation with $|V_{d-1}| = 1$ gives a lower bound, which can possibly be improved later by adding more vertices in V_{d-1}. In each iteration, sets V_d and V_{d-1} are updated, and the Iterative Formulation is solved. The Iterative Method summarized in Algorithm 2 solves the Iterative Formulation successively by updating the upper bounds until the upper bound is equal to the lower bound obtained by the best feasible solution. Let $N = 2m+1$ be the number of vertices in what follows.

Algorithm 2 Iterative Method

Input: $V_{d-1} = \emptyset$, $V_d = V$
$LB = 0$, $UB = \lfloor ((Nd)/2 \rfloor$
while $UB > LB$ **do**
  Solve Iterative Formulation D with V_d and V_{d-1}.
  if D is feasible **then**
    $LB = \max(z^*, LB)$ where z^* is the optimal objective function value of D
  end if
  Pick $u \in V_d$
  $V_d = V_d \setminus \{u\}$
  $V_{d-1} = V_{d-1} \cup \{u\}$
  $UB = \max(\lfloor ((Nd)/2 - 0.5|V_{d-1}| \rfloor, LB)$
end while
return UB and the optimal solution of the Iterative Formulation giving the best LB.

Our Iterative Method does not primarily focus on reducing symmetry, instead, it focuses on reducing the size of the feasible region. In the first iteration, where $V_{d-1} = \emptyset$, all variables are equivalent. Still, in each iteration, one of the degree constraints for a vertex changes, reducing the symmetry. To reduce the symmetry further, we can order the vertices of V_{d-1} lexicographically so that their degrees are non-increasing.

3.3 Iterative Method with Orbital Branching

Orbital Branching exploits the symmetry for branching decisions, and the Iterative Method reduces the size of the feasible region while keeping most of the symmetry intact. Besides, the Orbital Branching interferes with the default branch-and-cut decisions of the solver while the Iterative Method modifies the problem formulation. Therefore it is possible to use these methods jointly so that the same iterations as in the Iterative Method are done, but branching decisions are made with Orbital Branching. The Iterative Formulation is less symmetric than the Basic Formulation since Equations (17) are equalities.

4 Computational Experiments

Computations have been conducted on Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz with 32 GB RAM using CPLEX 20.1.0 on 10 threads. We used CPLEX 20.1.0 with C++ to implement the methods. We enforce a limit of 1800 seconds for each run. In the following tables that show our computational results, "PreUB" stands for the precomputed upper bound, "LB" stands for the lower bound found by CPLEX, "UB" stands for the upper bound found by CPLEX, "Gap" represents the relative optimality gap, "Time" is the execution time, and "Node" is the number of nodes in the branch-and-cut tree. In Tables 1,2, and 3, lines corresponding to optimality are marked with a dark background. We tested instances with $7 \leq d \leq 13$ and $d < m \leq Z(d)$. When d is odd, $Z(d)$ is not known beforehand. However, by definition of $Z(d)$, a graph with matching number $m = Z(d)$ is almost d-regular. Accordingly, we find $Z(d)$ based on the degrees in the resulting edge-extremal graph. More precisely, $Z(d)$ is the value of the matching number m for which the LB is equal to the precomputed UB and the resulting extremal graph is factor-critical, which we check by computation. When d is even, we have $Z(d) = \lfloor 5d/4 \rfloor$ by Lemma 2.2. This implies $Z(8) = 10, Z(10) = 12$ and $Z(12) = 15$. Then, it follows from Theorem 2.2 that $f_{\Delta}(8, 10) = 84$, $f_{\Delta}(10, 12) = 125$ and $f_{\Delta}(12, 15) = 186$. From the computations, we learned the $Z(d)$ values for the new d values 7, 9, 11, and 13; namely $Z(7) = 9$, $Z(9) = 12$, $Z(11) = 15$, $Z(13) = 17$. Overall, if we check the solved instances, the edge counts align with the Conjecture 2.2 given in Ahanjideh, Ekim, and Yıldız [3].

For solving the Basic Formulation (5)-(8), we started with CPLEX with default parameters and CPLEX with symmetry breaking parameter set to its most aggressive setting (*IloCplex::Param::Preprocessing::Symmetry* = 5). The results are shared in Table 1. Both of the approaches solved only 7 of the instances to optimality. Overall, CPLEX with symmetry breaking is slightly better due to a lower average time. Gaps are equivalent for both of the approaches. Note that the upper bound levels are equivalent to precomputed upper bound values that can be achieved by the maximum degree constraint. For example, when d is 7 and m is 8, there are 17 vertices since we can assume that an extremal graph is factor-critical. Since d is odd, the maximum edge count is achieved via an almost d-regular graph (the sum of all the degrees should be even). 16 vertices with degree 7 and one vertex with degree 6, yielding 59 edges in total. This explains why the only solved instances are for $m = Z(d)$ values. When $m = Z(d)$, the resulting graph is d-regular or almost d-regular, which has an equivalent edge count to the bound we discussed. From an inspection of the results, we can conclude that our initial approaches are not good enough for decreasing the upper bound found by CPLEX, but they are promising for finding feasible solutions. This might be due to symmetry. In our next experiment, we test the efficacy of using Orbital Branching with Basic Formulation (Subsection 3.1).

We implemented Orbital Branching using the callback mechanism of CPLEX. For finding orbits, we used nauty 2.7r3, a software library for computing automorphism groups of graphs by McKay and Piperno [15]. At each branch-and-cut node, we find the variables set to 0 and 1 from CPLEX and update the constraints accordingly. Then, we construct the updated bipartite graph using binding constraints. The resulting graph (and coloring) is sent to nauty. For calculating orbits, nauty calculates the automorphism groups utilizing graph isomorphism problem, which is a $GI-complete$ problem [16]. This implies that the problem of calculating orbits is neither known to be NP-complete nor known to be polynomial-time solvable. For large instances, run time can be as high as 150 seconds, while it is around 0.1 seconds for small instances. This can be seen in Figure 4, where instance 7_8 means instance with $d = 7$ and $m = 8$. Since there can be tens of thousands of nodes in a branch-and-cut tree, even a runtime of one second for orbit calculation results in an excessive CPU time. Therefore, we analyze the average orbit size for each depth in a branch-and-cut tree. Orbits tend to get smaller as we go deeper in the branch-and-cut tree, which can be seen from Figure 5. For example, using Orbital Branching after depth 20 only adds overhead to the instance with $d = 7$ and $m = 8$. Due to this, we run Orbital Branching until a certain depth. This depth is initialized as a large number in the beginning. When we find a node with orbit size 1, the depth of that node becomes the limit; for deeper nodes, orbits are not calculated. After that depth, the default CPLEX branching strategy is used. Branching is done on the orbit with the maximum number of vertices since it sets the values for more variables in the child nodes. If the largest orbit only has a single variable, we use the default CPLEX branching strategy.

Parameters			Basic Formulation					Basic Formulation + Symmetry Breaking				
d	***m***	**PreUB**	**LB**	**UB**	**Gap**	**Time**	**Node**	**LB**	**UB**	**Gap**	**Time**	**Node**
7	8	59	58	59	1.72%	1800	8412700	58	59	1.72%	1800	4657959
7	9	66	66	66	0.00%	0	0	66	66	0.00%	0	0
8	9	76	74	76	2.70%	1800	4605513	74	76	2.70%	1800	3012473
8	10	84	84	84	0.00%	0	0	84	84	0.00%	0	0
9	10	94	92	94	2.17%	1800	2952258	92	94	2.17%	1800	1836200
9	11	103	102	103	0.98%	1800	1356502	102	103	0.98%	1800	1122091
9	12	112	112	112	0.00%	2	428	112	112	0.00%	3	465
10	11	115	112	115	2.68%	1800	2049077	112	115	2.68%	1800	1274473
10	12	125	125	125	0.00%	0	0	125	125	0.00%	0	0
11	12	137	134	137	2.24%	1800	530890	134	137	2.24%	1800	809227
11	13	148	146	148	1.37%	1800	293767	146	148	1.37%	1800	311019
11	14	159	158	159	0.63%	1800	257540	158	159	0.63%	1800	257540
11	15	170	170	170	0.00%	0	0	170	170	0.00%	0	0
12	13	162	158	162	2.53%	1800	471193	158	162	2.53%	1800	392793
12	14	174	171	174	1.75%	1800	271165	171	174	1.75%	1800	218547
12	15	186	186	186	0.00%	53	6077	186	186	0.00%	707	82348
13	14	188	184	188	2.17%	1800	326937	184	188	2.17%	1800	292989
13	15	201	198	201	1.52%	1800	265385	198	201	1.52%	1800	173732
13	16	214	212	214	0.94%	1800	148400	212	214	0.94%	1800	144746
13	17	227	227	227	0.00%	1200	45158	227	227	0.00%	21	311
Avg			138.45	140	1.12%	1233	1099650	138.45	140	1.12%	1207	729346

Table 1: Basic formulation performance summary.

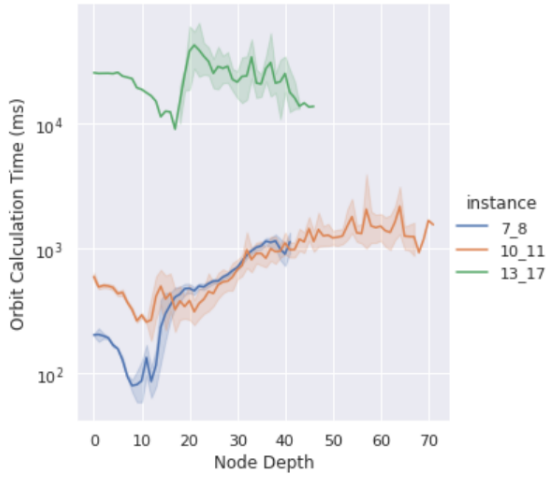

Figure 4: nauty runtime with depth.

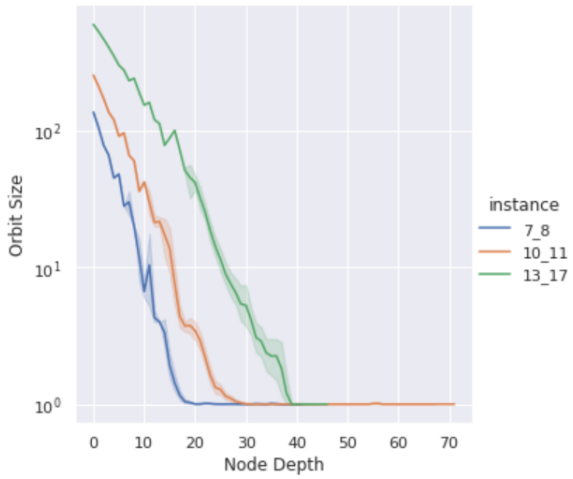

Figure 5: Orbit sizes with depth.

Parameters			Orbital Branching				
d	m	PreUB	LB	UB	Gap	Time	Node
7	8	59	58	58	0.00%	14	57945
7	9	66	66	66	0.00%	0	0
8	9	76	74	74	0.00%	61	236040
8	10	84	84	84	0.00%	0	0
9	10	94	92	92	0.00%	558	1781858
9	11	103	102	103	0.98%	1800	2651099
9	12	112	112	112	0.00%	21	193
10	11	115	112	114	1.79%	1800	1838556
10	12	125	125	125	0.00%	0	0
11	12	137	134	137	2.24%	1800	1223806
11	13	148	146	148	1.37%	1800	1142867
11	14	159	158	159	0.63%	1800	428121
11	15	170	170	170	0.00%	0	0
12	13	162	158	162	2.53%	1800	567720
12	14	174	171	174	1.75%	1800	444420
12	15	186	186	186	0.00%	361	4769
13	14	188	184	188	2.17%	1800	278258
13	15	201	198	201	1.52%	1800	85467
13	16	214	212	214	0.94%	1800	33132
13	17	227	225	227	0.89%	1800	2756
Avg			138.4	139.7	0.98%	1041	538850

Table 2: Orbital branching performance summary.

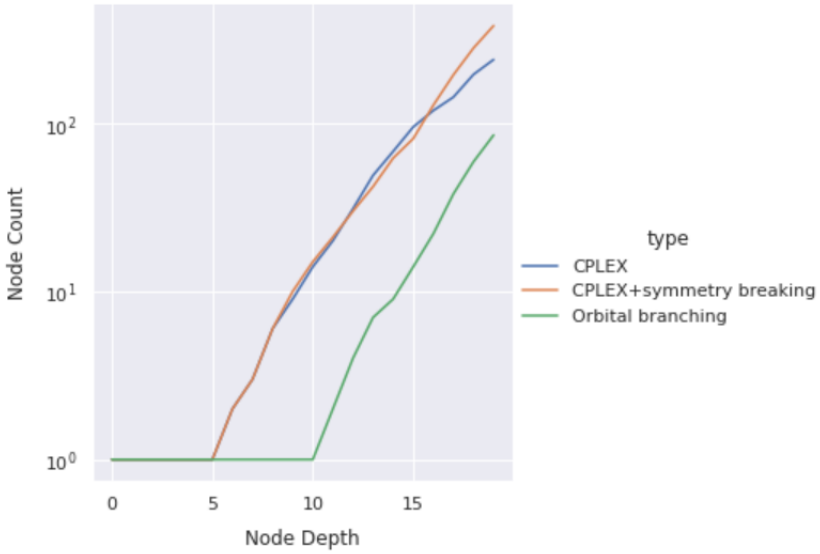

Figure 6: Branch-and-cut node count vs branch-and-cut node depth

The results for Orbital Branching are shared in Table 2. Orbital Branching solves 9 instances. Comparing Tables 1 and 2, we observe that while the Basic Formulation fails to find a solution to instances other than $m = Z(d)$, Orbital Branching solves all instances for $d = 7$ and $d = 8$. The difference can be attributed to how Orbital Branching prunes the initial nodes, which can be seen from Figure 6. Orbital Branching prunes all nodes but one until depth 10 for the instance with $d = 7$ and $m = 8$. This enables it to work on a significantly smaller feasible region. Yet, when the problem size increases, its performance deteriorates. For $d = 13$ and $m = 17$, it fails to find a provably optimal solution, while CPLEX with aggressive symmetry breaking finds in fewer nodes. Some difference is

Parameters			Iterative					Iterative + Orbital Branching				
d	m	PreUB	LB	UB	Gap	Time	Node	LB	UB	Gap	Time	Node
7	8	59	58	58	0.00%	0	145	58	58	0.00%	0	57
7	9	66	66	66	0.00%	0	0	66	66	0.00%	0	0
8	9	76	74	74	0.00%	3	12880	74	74	0.00%	4	2794
8	10	84	84	84	0.00%	0	0	84	84	0.00%	0	0
9	10	94	92	92	0.00%	2	5149	92	92	0.00%	11	1990
9	11	103	102	102	0.00%	5	12459	102	102	0.00%	36	4170
9	12	112	112	112	0.00%	1	33	112	112	0.00%	368	229
10	11	115	112	112	0.00%	100	208001	112	112	0.00%	79	10114
10	12	125	125	125	0.00%	1	0	125	125	0.00%	2	0
11	12	137	134	134	0.00%	176	253162	134	134	0.00%	146	15106
11	13	148	146	146	0.00%	206	362834	146	146	0.00%	355	371215
11	14	159	158	159	0.63%	1800	1087900	158	159	0.63%	1800	706832
11	15	170	170	170	0.00%	2	38	170	170	0.00%	1	0
12	13	162	158	159	0.63%	1800	2264281	158	158	0.00%	1623	2020924
12	14	174	171	172	0.58%	1800	1579790	171	172	0.58%	1800	1101510
12	15	186	186	186	0.00%	0	0	186	186	0.00%	0	0
13	14	188	184	185	0.54%	1800	1495811	184	185	0.54%	1800	2294463
13	15	201	198	200	1.01%	1800	1313790	198	200	1.01%	1800	935875
13	16	214	212	213	0.47%	1800	690328	212	213	0.47%	1800	577438
13	17	227	227	227	0.00%	56	5696	227	227	0.00%	72	543
Avg			138.45	138.8	0.25%	568	464615	138.45	138.75	0.22%	585	402163

Table 3: Iterative Formulation performance summary.

Method	Solved	Gap	Time	Node
Basic Formulation	7	1.12%	1233	1099650
Basic Formulation + Symmetry Breaking	7	1.12%	1207	729346
Orbital Branching	9	0,98%	1041	538850
Iterative Method	14	0,25%	568	464615
Iterative Method + Orbital Branching	15	0.22%	585	402163

Table 4: Performance summary of all methods.

due to nauty iterations running over 50 seconds. Yet, for all methods, most instances are unsolved, and the problem of decreasing the upper bound continues.

In our next experiment, we explore the performance of Iterative Methods (Subsections 3.2 and 3.3). In the Iterative Method, we use the global callback mechanism of CPLEX. Instead of waiting for an iteration to end, we use the lower bound information from previous iterations to stop the current iteration if needed. We share the results for both the Iterative Method and the Iterative Method with Orbital Branching in Table 3. The Iterative Method optimally solves 14 instances, and the Iterative Method with Orbital Branching solves one more instance to optimality. These numbers are significantly higher than methods using Basic Formulation. The Iterative Method has slightly less run time, while the combined method solves the problem using significantly fewer nodes.

The overall performance summary is shared in Table 4. We can see an increase in instances solved as we use more elaborate approaches. The number of solved instances for the Basic Formulation is 7, but all of them are for $m = Z(d)$ values; therefore, there is a significant increase in the performance with the new approaches. Especially, both Iterative Methods are significantly better than the others.

It is important to note that Iterative Methods do not only solve some new instances that support Conjecture 2.1, but also provide us with graphs that follow a pattern. This trend allows us to suggest a general construction for graphs claimed to be extremal in Conjecture 2.1 for all maximum degree d and matching number i with $7 \leq d < i < Z(d)$. For $0 \leq t < Z(d) - d$, let us describe the graph $B_{d,d+t}$, where $i = d + t$ as follows (see Figure 4): take a complete bipartite graph $K_{d+t,d+t}$ with $d + t$ vertices in each side. Consider $d - 1$ vertices of each side (the set denoted F in Figure 4) and remove t disjoint perfect matchings between them. Note that this is possible because since $t < Z(d) - d$ and $Z(d) \leq \lceil 5(d+1)/4 \rceil$ by Lemma 2.2, we have $t < \lceil (d+1)/4 \rceil + 1 < d$. Then, consider the remaining $t + 1$ vertices of each side (the set denoted H in Figure 4), remove all edges between, introduce one

additional vertex v, and make it adjacent to all the $t+1$ vertices of each side. With this construction, all the vertices except v have degree d. Let us remark that for $t = 0$, the graph $B_{d,d+t}$ corresponds to the graph A_d given in [3] as an extremal graph for the case $d = m$ with $f_\Delta(d, d) = d^2 + 1$ edges. The graph A_d, or equivalently $B_{d,d}$, can be seen as a 5-cycle whose two adjacent vertices are replaced with independent sets of size $d-1$ each and edges between these sets and other vertices (or set) are replaced with complete links. From this perspective, our construction of the graph $B_{d,d+t}$ can be seen as a generalization of A_d for matching numbers between $d+1$ and $Z(d)-1$.

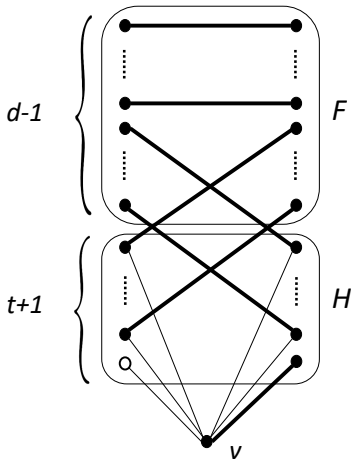

Figure 7: The graph $B_{d,d+t}$ where the set H induces an independent set and the set F induces a complete bipartite graph from which t distinct perfect matchings are removed. The deletion of the vertex v leaves a bipartite graph that has complete links between the opposite parts of the sets F and H. Bold edges show a maximum matching of size $d+t$.

Proposition 4.1. *For all $i = d+t$ such that $d < i < Z(d)$, the graph $B_{d,d+t}$ has $di+i-d+1$ edges, $\Delta(B_{d,d+t}) = d$ and $\nu(B_{d,d+t}) = d+t = i$; thus, it is an extremal graph if Conjecture 2.1 holds.*

Proof. First, we note that $t < Z(d) - d$ and thus the graph $B_{d,d+t}$ has one vertex of degree $2t+2$ (the additional vertex) and $2d+2t$ vertices each one of degree d. Since $t < \lceil (d+1)/4 \rceil + 1$, we have $2t+2 \leq d$ for $d \geq 7$, it follows that $\Delta(B_{d,d+t}) \leq d$. Moreover, we have $\nu(B_{d,d+t}) \leq d+t = i$ since it has $2d+2t+1$ vertices. More precisely a matching of size $d+t = i$ can be obtained as shown in Figure 4 by bold edges; set a matching of size $d-t-2$ in F, match v with one vertex in H, leave one vertex from the other side of H unmatched, now since $t+1 \leq d/2 < d-1$, all vertices of H but these two can be matched with the remaining vertices of F (recall that each side of F is completely joined to the opposite side of H). Now, counting the number of edges using the degrees yields $d^2+dt+t+1$ edges which is, for $i = d+t$, equal to $di+i-d+1$ as suggested in Conjecture 2.1. □

Finally, we use the above mentioned results to construct an extremal graphs for all d values $7, 8, 9, 10$ with the Knapsack Formulation. For these d values, there are at most 4 components in the Knapsack Formulation; therefore, even for $d = 10$ and $m = 10000$, it takes 0.2 seconds to obtain an optimal solution. The number of each extremal component in an optimal solution and resulting edge counts are shared in Table 5. In Table 5, edge counts are shared for $2d < m \leq 3d$. The column "*#star*" denotes the number of d-stars, "*#Z(d)*" denotes the number of graphs with matching number $Z(d)$, and "*#other*" denotes the number of other components. In Table 6, more extensive results are shared for $d = 8$ where "*comp_i*" denotes the number of components with matching number i in an extremal graph. When m increases by 10, the only difference is an additional component with matching number $Z(8) = 10$. Tables 5 and 6 show that Proposition 2.1, which is conditional to Conjecture 2.1, holds for $d = 7, 8, 9$ and 10. Proposition 2.1 describes a pattern on the construction of an edge-extremal graph; it suggests that there is an extremal graph having as many extremal components as possible with matching number $Z(d)$, completed with at most one extremal component with matching number between d and $Z(d)$, and d-stars; all of our results are compatible with this foresight and support

d	m	edge count	#d-star	#$Z(d)$	#other
7	15	108	6	1	0
7	16	116	0	1	1
7	17	124	0	1	1
7	18	132	0	2	0
7	19	139	1	2	0
7	20	146	2	2	0
7	21	153	3	2	0
8	17	140	7	1	0
8	18	148	8	1	0
8	19	158	0	1	1
8	20	168	0	2	0
8	21	176	1	2	0
8	22	184	2	2	0
8	23	192	3	2	0
8	24	200	4	2	0
9	19	175	7	1	0
9	20	184	8	1	0
9	21	194	0	1	1
9	22	204	0	1	1
9	23	214	0	1	1
9	24	224	0	2	0
9	25	233	1	2	0
9	26	242	2	2	0
9	27	251	3	2	0
10	21	215	9	1	0
10	22	226	0	1	1
10	23	237	0	1	1
10	24	250	0	2	0
10	25	260	1	2	0
10	26	270	2	2	0
10	27	280	3	2	0
10	28	290	4	2	0
10	29	300	5	2	0
10	30	310	6	2	0

Table 5: Knapsack Formulation results.

therefore both Conjecture 2.1 and 2.2. It follows that the edge numbers of extremal graphs for all m values and for $d \in \{7, 8, 9, 10\}$ are equal to the formula given in Conjecture 2.2.

d	m	edge count	d-star	comp_8	comp_9	comp_10
8	15	124	5	0	0	1
8	16	132	6	0	0	1
8	17	140	7	0	0	1
8	18	149	0	1	0	1
8	19	158	0	0	1	1
8	20	168	0	0	0	2
8	21	176	1	0	0	2
8	22	184	2	0	0	2
8	23	192	3	0	0	2
8	24	200	4	0	0	2
8	25	208	5	0	0	2
8	26	216	6	0	0	2
8	27	224	7	0	0	2
8	28	233	0	1	0	2
8	29	242	0	0	1	2
8	30	252	0	0	0	3
8	31	260	1	0	0	3
8	32	268	2	0	0	3
8	33	276	3	0	0	3
8	34	284	4	0	0	3
8	35	292	5	0	0	3
8	36	300	6	0	0	3
8	37	308	7	0	0	3
8	38	317	0	1	0	3
8	39	326	0	0	1	3
8	40	336	0	0	0	4
8	41	344	1	0	0	4
8	42	352	2	0	0	4
8	43	360	3	0	0	4
8	44	368	4	0	0	4
8	45	376	5	0	0	4
8	46	384	6	0	0	4
8	47	392	7	0	0	4

Table 6: Knapsack Formulation results for $d = 8$.

5 Conclusion

In this paper, we studied the open cases for the problem of finding the maximum number of edges in a triangle-free graph with maximum degree at most d and matching number at most m. We suggested several integer programming methods, which is, to the best of our knowledge, a new approach for this extremal problem.

Since our Basic Formulation is highly symmetric, most of our efforts have been about breaking the symmetry. We proposed two different approaches, first exploiting symmetry for branching decisions using Orbital Branching and then an Iterative Method exploiting the closeness of the precomputed upper bound and the optimal solution value. As we used a combination of the Iterative Method and Orbital Branching, we observed an increase in the number of instances optimally solved and an overall decrease in solution time. Our approach extends the known cases from $d \leq 6$ in [3] to $d \leq 10$ (both for all $m > d$). We could also identify some extremal components for $d = 11, 12$ and 13. The only missing extremal component for $d = 11$ is for $m = 14$; if this extremal component is found then the Knapsack Formulation can compute all extremal graphs for $d = 11$ and any $m > d$.

It is important to note that although integer programming approaches will be limited in finding extremal components for higher d and m values, they give us more evidence on Conjectures 2.1, 2.2 and Proposition 2.1. Indeed, all our findings support the claim that there is an extremal triangle-free

graph with as many factor-critical extremal components with matching number $Z(d)$ as possible , at most one factor-critical extremal component with matching number less than $Z(d)$, and some d-stars. Thus, our results provide additional motivation to search for a formal (structural) proof of Conjectures 2.1 and 2.2, which will most probably require more powerful tools.

Finally, we pose the following question. We note that our integer programming formulations exploit the fact that extremal components are factor-critical by using the implied vertex number in their formulations. However, they do not explicitly force the constructed graphs to be factor-critical. Clearly, a factor-critical graph with matching number m has $2m+1$ vertices, but a graph with $2m+1$ vertices is not necessarily factor-critical. However, it turns out that all extremal components resulting from our integer programming formulations are factor-critical. This observation suggests that it might be interesting to investigate the following question: is it true that a triangle-free extremal graph G with matching number m and maximum degree d such that $d < m < Z(d)$ and having $2m+1$ vertices is factor-critical?

Acknowledgments

This work has been supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the grant number 122M452.

References

[1] V. Chvátal and D. Hanson, "Degrees and matchings," *Journal of Combinatorial Theory, Series B*, vol. 20, no. 2, pp. 128–138, 1976.

[2] N. Balachandran and N. Khare, "Graphs with restricted valency and matching number," *Discrete Mathematics*, vol. 309, no. 12, pp. 4176–4180, 2009.

[3] M. Ahanjideh, T. Ekim, and M. A. Yıldız, "Maximum size of a triangle-free graph with bounded maximum degree and matching number," arXiv, preprint, 2022. arXiv: 2207.02271.

[4] P. Turan, "On an extremal problem in graph theory," *Középisk. Mat. és Fiz. Lapok*, vol. 48, pp. 436–452, 1941.

[5] P. Erdös and R. Rado, "Intersection theorems for systems of sets," *Journal of the London Mathematical Society*, vol. 1, no. 1, pp. 85–90, 1960.

[6] C. Dibek, T. Ekim, and P. Heggernes, "Maximum number of edges in claw-free graphs whose maximum degree and matching number are bounded," *Discrete Mathematics*, vol. 340, no. 5, pp. 927–934, 2017.

[7] J. Blair, P. Heggernes, P. Lima, and D. Lokshtanov, "On the maximum number of edges in chordal graphs of bounded degree and matching number," *Algorithmica*, vol. 84, pp. 3587–3602, 2022.

[8] E. K. Måland, "Maximum number of edges in graph classes under degree and matching constraints," M.S. thesis, The University of Bergen, 2015.

[9] F. Furini, I. Ljubić, and P. S. Segundo, "A new bilevel optimization approach for computing Ramsey numbers," *Manuscript*, 2021. [Online]. Available: https://optimization-online.org/?p=17323.

[10] P. Hansen, M. Aouchiche, G. Caporossi, A. Hertz, and C. Sellal, "Mixed integer programming and extremal chemical graphs," *Journal of Chemistry and Chemical Engineering Systems*, vol. 3, pp. 22–30, 2018.

[11] R. Ito, N. A. Azam, C. Wang, A. Shurbevski, H. Nagamochi, and T. Akutsu, *A novel method for the inverse QSAR/QSPR to monocyclic chemical compounds based on artificial neural networks and integer programming*. Cham: Springer, 2021, pp. 641–655.

[12] J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio, "Orbital branching," *Mathematical Programming*, vol. 126, no. 1, pp. 147–178, 2011.

[13] F. Margot, "Exploiting orbits in symmetric ILP," *Mathematical Programming*, vol. 98, no. 1, pp. 3–21, 2003.

[14] M. E. Pfetsch and T. Rehn, "A computational comparison of symmetry handling methods for mixed integer programs," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 37–93, 2019.

[15] B. D. McKay and A. Piperno, "Practical graph isomorphism, ii," *Journal of Symbolic Computation*, vol. 60, pp. 94–112, 2014.

[16] D. S. Johnson, "The NP-completeness column," *ACM Transactions on Algorithms (TALG)*, vol. 1, no. 1, pp. 160–176, 2005.