Edge Colouring

Student Name: Jun Eeo

Supervisor Name: Daniel Paulusma

Submitted as part of the degree of [BSc Computer Science] to the

Board of Examiners in the Department of Computer Sciences, Durham University

Abstract — **Context/Background -** Determining the optimal number of colours needed to edge colour a given graph (that is, to compute a mapping on the edges such that two adjacent edges do not get the same colour) is NP-complete in general, even though the bounds on the best (Δ) and worst ($\Delta + 1$) case number of colours needed are close to each other. There are existing works on classifying whether a graph needs Δ or $\Delta + 1$ colours for an edge-colouring based on the structure of the graph, e.g. whether it contains some sub-graph.

Aims - The aim of this project is to come up with some conjectures about substructures which are contained in graphs which need $\Delta + 1$ colours – although we note that such a task might be infeasible to complete, since the problem of recognising such graphs is NP-hard. We aim to investigate and perhaps find some new results in specific cases of existing, related conjectures for specific graph classes.

Method - We would implement a number of edge-colouring heuristics, evaluate the performance of our implementations, and then use them to perform computational experiments. Since the number of graphs of size n grows extremely quickly, we have to resort to edge-colouring randomly generated graphs with certain structural properties.

Results - Using our implementations of the Vizing Heuristic, we are able to derive new conjectures from results of our computational experiments. Furthermore, we were able to improve the practical performance of one heuristic (the Counting-Based Heuristic), and find new, optimal edge-colourings for some benchmark graphs.

Conclusions - We are able to obtain new conjectures (from our computational experiments) in cases of $\Delta \geq 5$ of the Hilton-Zhao Conjecture (which exhibits interesting structural properties), as well as the Bad Cores problem defined in our project. Computational experiments such as ours may be used in the future as starting points for further investigations into problems.

Keywords — Edge colouring, graph theory, Hilton-Zhao conjecture, Bad Cores, Heuristics

I INTRODUCTION

An Edge Colouring φ of a graph G is a mapping $E(G) \to \mathcal{C}$ such that $\varphi(e) \neq \varphi(f)$ whenever the two edges are adjacent. It is important that we use as few colours as possible. One of the many applications of Edge Colouring is that of fixture scheduling (Skiena 2012). For example, we might be running a football league where teams play one game a week. We then want to compute a schedule such that each time can complete their required games with the minimum number of weeks required for the league.

To do that, we construct the *game graph*, with a vertex for each team, and edges between them denoting a game that needs to be played (for simplicity, we assume that a team only plays another team at most once). An edge-colouring of this graph will then give us a schedule with the games in week 1 being the edges coloured c_1, week 2 with c_2, and so on. It is clear that in this case, we would want the schedule to use the minimum number of colours (weeks required).

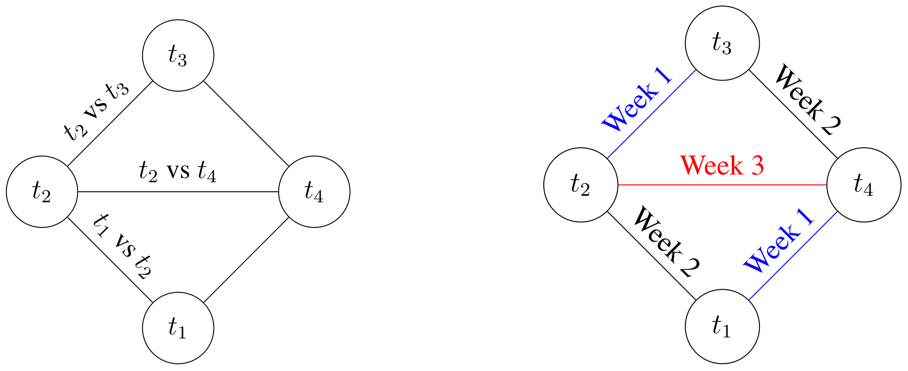


Figure 1: Game graph for 4 teams, and a corresponding schedule from an edge-colouring.

A Formalisation & Definitions

In our project we will only consider undirected simple graphs – graphs $G = (V, E)$ such that there are no self-loops or multiple edges. An *edge colouring* of such a graph is the mapping $\varphi : E \mapsto \mathcal{C}$ from the set of edges to a set of colours, such that whenever edges uv and uw are adjacent (with $v \neq w$), $\varphi(uv) \neq \varphi(uw)$. A *partial* edge colouring is one that is not defined for all edges; this concept will be useful in the design of edge-colouring heuristics.

In general, finding an *optimal edge-colouring* – one that uses the smallest possible set of colours, is NP-Hard (Holyer 1981). We let the *edge-chromatic index*, $\chi'(G)$ denote the number of colours used by an optimal edge-colouring of G. Although it is NP-hard to compute $\chi'(G)$, Vizing (1964) showed that any simple graph G satisfies $\Delta \leq \chi'(G) \leq \Delta + 1$ – this result is known as *Vizing's Theorem*. Δ is defined as follows; we define the degree of a vertex $d(v) = |\{uv : uv \in E\}|$, i.e. the number of edges adjacent to it, and $\Delta = \max_{v \in V} d(v)$, the maximum degree of a graph. A graph G is *Class 1* if $\chi'(G) = \Delta$, otherwise it is *Class 2* (if $\chi'(G) = \Delta + 1$). Hoyler (1981) showed that determining whether a graph is Class 1 or 2 (the *Graph Classification Problem*) is NP-hard in general.

Let $Z \subseteq V$ denote a set of vertices. We denote the *induced subgraph* of Z on G by $G[Z] = (Z, \{uv : uv \in E \wedge u, v \in Z\})$. We say that a graph G is *H-free* if it does not contain H as an induced subgraph. The set of vertices of G with degree Δ is denoted by Λ_G, and the *core* of G is $G[\Lambda_G]$ (the subgraph formed by deleting all vertices with $d(v) < \Delta$ from the original graph). The *semicore* of a graph is given by $G[N(\Lambda_G)]$ where $N(Z) = Z \cup \{uv : uv \in E, v \in Z\}$.

The *Random Graph Model*, also known as the *Erdős-Renyi Graph Model* (Erdös et al. 1959) is defined as follows: $\mathcal{G}(n, p)$ is the distribution of random graphs with size (number of vertices) n, and probability p of there being an edge between any two distinct vertices. For our project, the theorem below is of relevance, especially in computational experiments:

Theorem 1 (Erdős & Wilson, 1977) *For a graph $G \sim \mathcal{G}(n, \frac{1}{2})$, the probability that G is Class 1 tends to 1 as $n \to \infty$.*

We additionally give the definitions of a relevant conjecture for our project:

Conjecture 1 (Hilton & Zhao, 1996) *All graphs which have $\Delta(G[\Lambda_G]) = 2$, and $\Delta(G) \geq 4$ are Class 2, if and only if they are overfull: $|E| > \Delta\lfloor |V|/2 \rfloor$.*

We note that one side of Hilton-Zhao Conjecture is a well-known result – any overfull graph is necessarily Class 2. In our project we define a *Bad Core* – we say that graph H is a bad core, if and only if there exists some graph G, with $G[\Lambda_G] = H$ and G is *underfull*: $|E| \leq \Delta\lfloor|V|/2\rfloor$. We refer to the vertices and edges that are added to H to produce G as the *Bad Extension* to H.

When presenting results from our investigations, we commonly use the disjoint union and join notation. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two *vertex-disjoint* graphs, i.e. $V_1 \cap V_2 = \emptyset$. The *join* is defined as $G_1 \times G_2 = (V_1 \cup V_2, E_1 \cup E_2 \cup (V_1 \times V_2))$, and the *union* is $G_1 + G_2 = (V_1 \cup V_2, E_1 \cup E_2)$. We write nG_1 to mean the union of n-many G_1s (with different vertices).

B Project Objectives and Achievement

We seek to answer the question, ”Do Class 2 Graphs have any common substructures?”, by constraining the structure of the graphs we investigate. We approach this problem computationally by splitting it into the deliverables given below:

Basic

- **B1** Implement the Misra & Gries Heuristic (Misra & Gries 1992).
- **B2** Implement the Vizing Heuristic (Januario 2012).
- **B3** Verify and replicate benchmark results in Januario T. (2012), and compare the performance of implementations **B1** and **B2**.
- **B4** Empirically confirm results on random graphs proven by Erdős & Wilson (1977).

Intermediate

- **I1** Investigate cases where $\Delta \geq 5$ for the Hilton-Zhao Conjecture.
- **I2** Investigate the Bad Cores Problem – try to find and classify Bad Cores.
- **I3** Implement the Counting-Based Heuristic from Ehrenfeucht et al. (1984)
- **I4** Compare the performance of **I3** against **B2**.

Advanced

- **A1** Use Hamilton to perform larger and more experiments.
- **A2** Analyse results from **I1**, **I2**, and **A3**, and formulate conjectures.
- **A3** Run implementation of heuristics on large benchmark graphs.

II RELATED WORK

A Results on Various Graph Classes

Despite being an NP-hard problem in general, the Classification problem has known results for some graph classes. Many graph classes admit a polynomial-time algorithm to compute an optimal edge colouring; for bipartite graphs, graphs whose vertex set V can be partitioned into two disjoint sets A and B such that $V = A \cup B$, $A \cap B = \emptyset$, there is König's Theorem, which states that every bipartite graph is Class 1. Behzad et al. (1967) used a simple construction with addition of edges modulo n to show that the edge chromatic index of the complete graph on n vertices, $K_n = (\{1, \ldots, n\}, \{uv : u \neq v, 1 \leq u, v \leq n\})$ is:

$$\chi'(K_n) = \begin{cases} n - 1 = \Delta & \text{if } n \text{ is even or } n = 1 \\ n = \Delta + 1 & \text{if } n \geq 3 \end{cases}$$

However there are many graph classes which do not have a polynomial-time algorithm for computing an optimal edge colouring (Cao et al. 2019) — one of the more surprising ones being P_4-free graphs (cographs), which are easily vertex-coloured. In a series of papers investigating the Graph Classification Problem, Vizing (1965) showed that the core of any Class 2 graph has ≥ 3 vertices. The proof of Theorem 1 is based on this observation and counting the number of vertices which have maximum degree.

Another result from Fournier (1977) shows that if the core, $G[\Lambda_G]$ has no cycles, then G is Class 1 — this is consistent with Vizing's earlier result that a Class 2 graph has a core containing ≥ 3 vertices, since we cannot form a cycle with only two vertices. A more recent result from Machado and de Figueiredo (2010) builds on the previous result, and shows that the edge-chromatic index of any graph is equal to the edge-chromatic index of its semicore. These results would benefit our project, since we can now constrain our search to connected graphs which are valid semicores.

A recent result from Craston & Rabern (2017) proved the case of $\Delta = 4$ for the Hilton-Zhao conjecture. They found that the only *connected* graph is $K_5 - e$, the complete graph on 5 vertices minus any one edge. Thus, a natural extension in this direction is to try to find similar results when we increase $\Delta \geq 5$ – are there similar results to be found? Can we find all the *connected* graphs that are Class 2? Does the conjecture still hold for larger values of Δ?

An older result from Chetwynd and Hilton (1990) shows that a graph with a core of size 3 is Class 2 if it has the following degree sequence: $((2m-2)^{2m-1}, 2m^3)$, for some $m \in \mathbb{Z}^+$. We say that a graph has degree sequence $(d_1^{n_1}, d_2^{n_2}, \ldots, d_m^{n_m})$ if it has exactly n_i vertices with degree d_i, for all $1 \leq n \leq m$. This result, as well as other known results for all the graph classes above is of particular relevance to us, since it allows us to test the colouring performance of our implementations.

B Edge Colouring Heuristics

Because Edge-Colouring is an NP-hard problem, any algorithm for edge-colouring a general graph is a heuristic algorithm. We classify them into two categories, those which satisfy Vizing's Theorem and have an upper bound on the worse-case number of colours used of $\Delta + 1$, and those which do not.

Perhaps the simplest approach would be using a greedy method – we build up a (partial) mapping φ edge-by-edge as follows: given some edge uv, pick the smallest colour available that is not used at u or v (we say a colour $c \in \mathcal{C}$ is *used* at vertex u if there is some edge uw with $\varphi(uw) = c$). This heuristic uses (in the worse case) $\mathcal{O}(|E|)$ colours – we can construct a bad ordering of edges, such that at each extension of φ, we always use the next unused colour. If we are 'lucky' however, we could use as little as Δ colours.

Heuristics belonging in the first category (realizing the upper bound in Vizing's Theorem) include the Misra & Gries Heuristic (Misra & Gries 1992), which is initially given as a constructive proof of Vizing's Theorem. The technique used is based on Colouring Fans and Kempe Exchanges, which are further elaborated in our Solution section. There is also the Vizing Heuristic from Januario, T. (2012) which is a modification of the Misra & Gries Heuristic.

Heuristics in the second category include meta-heuristic approaches. For instance, Khuri et al. (2000) used a genetic algorithm with the chromosome representation being a vector v of size $|E|$, where the i-th entry in the vector, v_i denotes the colour of the i-th edge (under some arbitrary ordering, e.g. lexicographical), and the crossover operators based on grouping scheme, where edges belonging to each colour are exchanged when two parents mate.

An interesting approach by Enochs and Wainwright (2001) is based on the simulated-annealing teachnique. The solution representation is similar to that of Khuri et al. (2000) (a vector of colours for each edge). To get from some solution v to a candidate solution v', they perturb v by randomly selecting some edge which is invalidly coloured, and attempts to assign a 'good' colour to it.

To avoid being stuck in local maxima, they further use a 'kick' function which randomly assigns a new colour to a some randomly selected edge that had an invalid colour. Their distance function is given below, and intuitively, measures the distance between the current solution and some feasible (valid) solution (where $e(v)$ denotes the number of invalidly coloured edges adjacent to v):

$$\sum_{v \in V} d(v) - e(v)$$

The main disadvantage of meta-heuristics is their comparably higher running time, making them impractical for use in our style of computational experiments where we need to scan through large numbers of graphs. Moreover, the Vizing Heuristic performs as good as the slower meta-heuristics on small graphs with $n < 200$ (Januario 2012), and has the benefit of having much lower running time.

There also exists a few simple and fast edge colouring heuristics proposed by (Hilgemeier et al. 2003) for large ($n > 500$) graphs, at the cost of not guaranteeing the worse case $\Delta + 1$ colours. Since most of our experiments wouldn't exceed ($n \approx 100$), they aren't relevant to our project except in the case of performance (in terms of time or colours used) benchmarks of our implementations.

C Graph Generation

There are algorithms to quickly (in terms of performance, not the asymptotic complexity) enumerate all non-isomorphic graphs of a given (small) size, as well as check if two graphs are isomorphic. Graphs G and H are **isomorphic** if there exists a mapping $f : V(G) \mapsto V(H)$, such that $uv \in E(G) \implies f(u)f(v) \in E(H)$. One particular implementation that we will use in our project is the `nauty` suite of tools from McKay and Piperno (2014), which includes well tested and highly efficient implementations of both such algorithms.

Because of the exponentially increasing number of graphs of size n in n, for large n, it will be infeasible for us to enumerate all of them. As such we would need to be able to randomly generate graphs. One such model of random graphs is the Binomial Graph model introduced earlier – $\mathcal{G}(n, p)$.

Another random graph model is the Barabási-Albert Model (Barabási & Albert 1999) can be used to generate scale-free graphs — graphs which degree distribution asymptotically follows a power law, i.e. the fraction of vertices $P(k)$ having k edges is $P(k) = \mathcal{O}(k^{-\gamma})$ for some $\gamma > 1$. Given 3 parameters n, m_0, m with $n \geq m_0 \geq m$, we generate a graph from the distribution as follows: we start with some initial network with m_0 vertices, and we add $n - m_0$ vertices one by one. Each new vertex being added is attached to m random existing vertices. The probability that the new vertex is connected to some existing vertex u is given by $p(u) = d(u)/\sum_{j \in V} d(j)$ – in other words, new vertices are more 'attracted' to existing vertices with higher degree.

Alternatively, there are graph generators based on degree sequences; given a *feasible* degree sequence – one that can be realized by some graph (i.e. there exists some graph having the given degree sequence), such an algorithm would randomly generate a graph that with the given degree sequence. This is also known as the Graph Realization problem. One such algorithm is the Havel-Hakimi algorithm from (Hakimi 1962), which is deterministic. We will use an algorithm from Bayati et al. (2010), implemented in the `networkx` Python library.

To the best of our knowledge, there exists no algorithms to generate graphs that do/do not contain arbitrary substructures which are relevant to us – for instance, those with some predefined core H, or those that are underfull).

III SOLUTION

For ease of development and debugging we first implemented our algorithms in Python. We then re-implemented them in the Go programming language to perform experiments locally, due to previous experience and familiarity with the language and the Python implementations being slow. For running experiments on Hamilton, we implemented the Vizing Heuristic, as well as some graph generators in C.

A Edge Colouring Heuristics

One common approach in the heuristics we implement is building up the mapping φ edge-by-edge. During execution, sometimes we may be unable to give some edge uv a colour, for example because there are no common free colours at both endpoints of the vertex, i.e. the set of unusued colours at u (the set of colours not used by any edge adjacent to u) and v are disjoint. In that case, we perform a *Kempe Exchange* – we pick some $\alpha \in \mathcal{C}$ that is unused at u, and some $\beta \in \mathcal{C}$ unused at v. We let $G_\varphi[\alpha, \beta]$ denote the subgraph induced by vertices with edges coloured either α or β by φ. Each component of $G_\varphi[\alpha, \beta]$ (if φ is a valid (partial)-colouring) is either a path or an even-cycle.

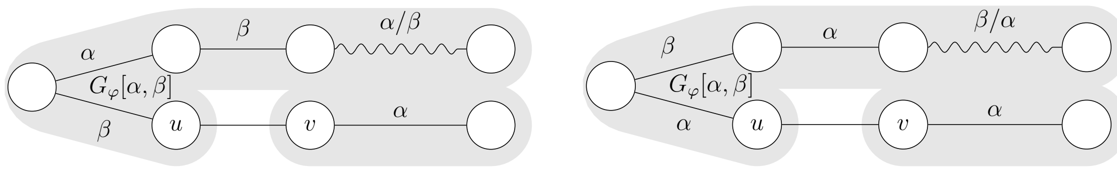


Figure 2: Before (left) and after (right) a Kempe Exchange.

Since both vertices u and v do not have unused colours in common, the vertex u lies on some path P in $G_\varphi[\alpha, \beta]$. We call this the $\alpha\beta$ path in G starting from u. By exchanging the colours along P, we free up β to be used at u, and we can give the colour β to edge uv (assuming that P does not end in v), while mainitaining the validity of φ.

In all implementations, we used an adjacency matrix to represent the graph and store the edge colours, since we found that this is more performant and easier to implement compared to the adjacency-list. In our pseudocode, we represent the colours used by integers for simplicity.

A.1 Misra & Gries Heuristic (B1)

The Misra & Gries Heuristic (Misra & Gries 1992) uses the concept of colouring fans. A colouring fan for an edge uv is a sequence $(v_1, v_2, \ldots, v_k)$ of distinct neighbours of u, such that $v_1 = v$ is uncoloured, and the colour $\alpha_i = \varphi(uv_{i+1})$ is unused by v_i, for all $1 \leq i < k$. We *rotate* the fan by giving uv_i the colour α_i, and uncolouring the last edge uv_k. This operation leaves φ valid, since for each i, α_i was free on v_i. We illustrate this procedure in Figure 3.

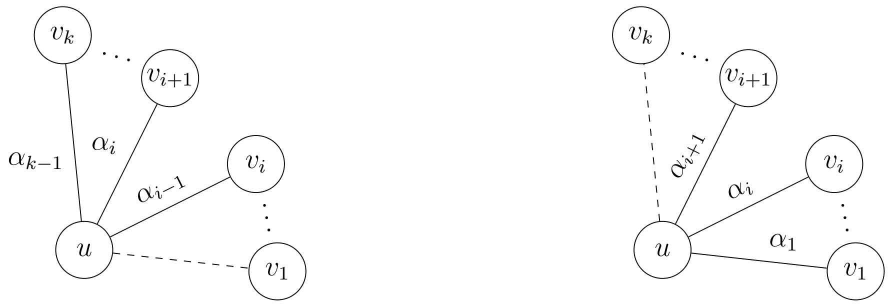


Figure 3: Before and after a fan rotation. The dashed line denotes an uncoloured edge.

Algorithm 1: Misra Gries Heuristic

```
Input: Graph $G = (V, E)$
while $\exists$ uncoloured $uv \in E$ do
    $(t_1, \ldots, t_n) \leftarrow MaximalFan(uv)$
    Let $\alpha \in Free(u)$
    Let $\beta \in Free(t_n)$
    Exchange the colours along the $\alpha\beta$-path from $u$ to $t_n$
    Let $w$ be such that $(t_1, \ldots, w)$ is a fan, and $\beta$ is free on $w$
    Rotate $(t_1, \ldots, w)$
    $\varphi(uw) \leftarrow \beta$
return $\varphi$
```

In the pseudo-code presented in Algorithm 1, we use $Free(u)$ to denote the set of colours unused by vertex u. This is maintained throughout the execution of the algorithm – whenever we colour the edge uv, $Free(u)$ and $Free(v)$ need to be updated appropriately. Initially, $Free(u)$ is set to $\{1, 2, \ldots, \Delta + 1\}$ for all vertices u.

To compute $MaximalFan(uv)$, we start from the fan (v), and extend it as follows: while there exists some neighbour t of u that is not in the fan, and $\varphi(ut)$ is a colour unused by the last vertex in the fan, add t to the fan. The running time of the algorithm is $\mathcal{O}(|V||E|) = \mathcal{O}(n^3)$. In practice, we gain some efficiency in some cases by immediately giving the edge uv a colour that is free at both endpoints if it is possible, thus avoiding computing the maximal fan.

A.2 Vizing Heuristic (B2)

The Vizing Heuristic (Januario 2012) is based on Vizing's Algorithm (Gould 1988) – we first use only Δ colours, rather than the full $\Delta + 1$ colours. Upon encountering an edge that cannot be coloured using only Δ colours, we increase the set of available colours to the full $\Delta + 1$ colours (i.e., we add $\Delta + 1$ to all $\mathit{Free}(u)$), and we continue execution as normal. Because we used this heuristic to get most of our results, we present the ideas behind, as well as a proof of correctness of the original algorithm that it's based on below.

Consider colouring some edge wv_0 (we are allowed to use the full $\Delta + 1$ colours) – since both w and v_0 have at most $\Delta - 1$ incident colour edges, there are at least two colours that are unused by both of them. If $\mathit{Free}(w) \cap \mathit{Free}(v_0) \neq \emptyset$, we are done – we can set $\varphi(wv_0)$ to some colour from that intersection.

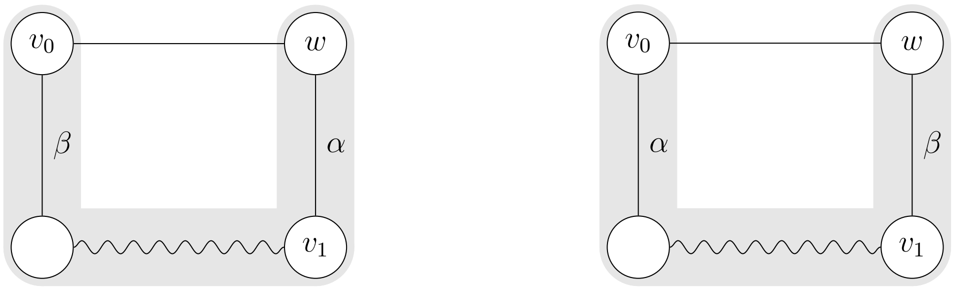


Figure 4: Before and after a Kempe Exchange when P ends in w.

However if that intersection is empty, let α be some colour free at v_0, and β some colour free at w. Consider the path P in $G_\varphi[\alpha, \beta]$ starting from v_0 (certainly, v_0 must be on the start of a path and not a cycle, since otherwise would imply that v_0 has some edge coloured α, a contradiction) – if it ends at w, then we are done; a simple Kempe Exchange frees up β at v_0. If P does end in in w, then β will be free at v_0, but β will not be free at w, since it had some adjacent edge (let's call it wv_1) with colour α beforehand which is now coloured β. We uncolour wv_1 and set $\varphi(wv_0) \leftarrow \beta$, and then colour wv_1 using the same procedure – remembering that we should avoid giving wv_1 the colour α. The rest of the proof in Gould (1988), shows that this procedure eventually terminates – and we get a $\mathcal{O}(\Delta|V||E|)$ time heuristic.

In the Vizing Heuristic, we initially only use Δ colours – so the first assumption that we always some unused colour at both vertices does not always hold. Thus, when we meet this situation we simply make the $\Delta + 1$-th colour available at every vertex. The pseudo-code for the Vizing Heuristic is given in Algorithm 2. In the pseudo-code, we remember not to give the edge wv_1 the same colour by storing it in the variable taboo.

Our initial Python implementations of the Vizing Heuristic and the Misra & Gries Heuristic suggest that the Vizing Heuristic has a slightly lower runtime and finds the optimal colouring more frequently compared to the Misra & Gries Heuristic – our findings match those of Januario (2012). In our Go and C implementations, we maintain $\mathit{Free}(v)$ as a bitset (a bit-vector where the presence of the i-th element is signaled by the i-th bit being zero or one).

We have two versions of the Vizing Heuristic in C – one version, `xc` uses a single 64-bit integer bitset to represent the free sets at each vertex. A bitset is a bit-vector $\vec{b}$ representing the set $\{i : \vec{b}_i = 1\}$. This approach is highly performant, since we can exploit the fact that most set operations (e.g. computing the intersection and union) are 'constant time' and can be done in one bitwise operation – for instance, $A \cap B$ can be computed with `A && B`. For more complex operations like counting the number of elements in the set or getting the i-th element, we were also able to map those to only a few CPU instructions using e.g. `popcountl` and `ffsl`, `pdep_u64` respectively.

Algorithm 2: Vizing Heuristic

```
Input: Graph $G = (V, E)$
$taboo \leftarrow nil$
$Free(v) \leftarrow \{1, 2, \ldots, \Delta\} \forall v \in V$
while there are uncoloured edges in $G$ do
    if $taboo = nil$ then
        $wv_0 \leftarrow$ some uncoloured edge in $G$
    if $\exists c \in Free(w) \cap Free(v_0)$ then
        $\varphi(wv_0) \leftarrow c$
        $taboo \leftarrow nil$
    else
        if $Free(v_0) \setminus \{taboo\} = \emptyset$ then
            Add $\Delta + 1$ to every $Free(v), v \in V$
            $\varphi(wv_0) \leftarrow \Delta + 1$
            $taboo \leftarrow nil$
        else
            Let $\alpha \in Free(v_0) \setminus \{taboo\}$
            if $taboo = nil$ then
                Let $\beta \in Free(w)$
            Let $P$ be the maximal $\beta\alpha$-path starting from $v_0$
            if $P$ does not end with $w$ then
                Exchange the colours along $P$
                $\varphi(wv_0) \leftarrow \beta$
                $taboo \leftarrow nil$
            else
                Uncolour the last edge of $P$
                $\varphi(wv_0) \leftarrow \alpha$
                $wv_0 \leftarrow$ last edge of $P$
                $taboo \leftarrow \alpha$
return $\varphi$
```

However, this approach places an upper bound on the maximum size of graphs that our algorithm can handle (64) – this wasn't an issue initially because most graphs of interest had small sizes. In order to enable us to do larger experiments, we built `lxc`, which is able to handle graphs of arbitrary size (constrained only by the amount of memory available) by using arrays of 64-bit integers for bitsets. For most of our experiments, it is sufficient (and more performant) to return an estimate of whether the input graph is Class 1 or 2 by returning early when we need to add the $\Delta + 1$-th colour (we run the full algorithm when benchmarking).

We find that we can improve colouring performance by sampling a random edge on each iteration. As shown in Figure 5, with naïve edge-sampling we run risk of an expensive cache-miss each iteration, since we are likely to make big 'jumps' in the memory. Thus we have to use a more *cache-aware* approach – with probability $7/8$ (chosen empirically) we choose an edge that is in the same 'region', otherwise we resort to naïve random sampling. We find that without our cache-aware technique, random sampling of edges is extremely slow for larger graphs. We also run the heuristic multiple times (and thus obtain different edge-orderings), stopping early

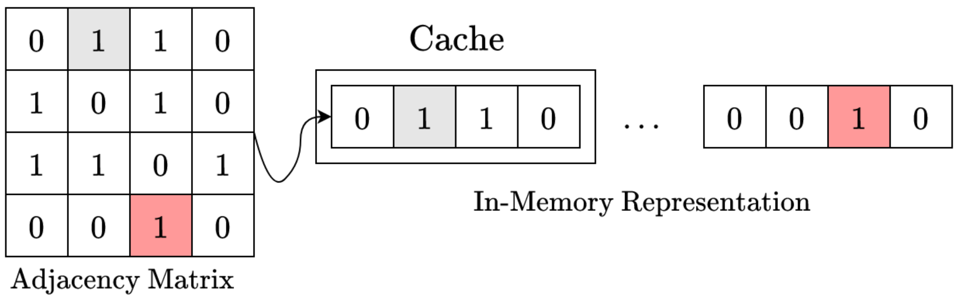


Figure 5: Locality issues with naïve edge sampling; the edge in gray is chosen in the previous iteration, and the edge in red is the current edge chosen — we get a cache miss.

when we classify a graph as Class 1, otherwise classifying it as Class 2. For most benchmark graphs we are able to consistently and quickly find the optimal colouring with our techniques.

A.3 Counting-Based Heuristic (I3)

Our implementation is based on a simplification presented in (Galby et al. 2019) of the original algorithm by (Ehrenfeucht et al. 1984) – a constructive proof of Vizing's Theorem based on a counting argument. The simplification was originally meant for computing a Δ-colouring of the graph G if some Δ colouring of the semi-core is given, but we managed to adapt it for colouring general graphs. In the pseudo-code, we let $F(uv)$ be the set of colours that edge uv can take; this is maintained throughout execution of the heuristic. To avoid storing $O(|V|^2)$ sets, a practical implementation would compute $F(uv)$ as required, as $\mathit{Free}(u) \cap \mathit{Free}(v)$.

Unlike the previous heuristics, we extend φ vertex-by-vertex. In the pseudo-code, we let Z be the set of vertices which are processed; at the end of the execution, we have $Z = V(G)$.

We made a small modification similar to that of the Vizing Heuristic – we first attempt to produce a colouring using only Δ colours, and failing that, continue execution using $\Delta + 1$ colours. We present the algorithm and our modifications highlighted in yellow in Algorithm 3.

We found that the original heuristic performs worse on most random graphs, but with our modification, we achieved similar or better (in particular, $\mathcal{G}(n, 0.95)$) colouring performance compared to the Vizing Heuristic. We did not implement this heuristic for running experiments on Hamilton, on the basis that our implementation of both modified and unmodified versions are around 6-7 times slower than the Vizing Heuristic; given the time taken to complete one execution of the Counting-Based heuristic, we can run the Vizing Heuristic on the same graph multiple times.

B Graph Generation

To complete deliverable **B4** we implemented a simple graph generator for the $\mathcal{G}(n, p)$ binomial graph model in Algorithm 4. We assume we have access to $\mathit{rand}()$, a function returning some real number in $[0, 1)$ with uniform probability).

For completing deliverables **I1** and **I2**, we made an algorithm to generate a valid semi-core of size $\in [m + 1, n]$, given a core H of size m, a maximum degree Δ, and an upper bound n. Our algorithm first builds up a list of allowed degrees (a *budget*) for each vertex v. For each vertex v

Algorithm 3: Modified Counting-Based Heuristic

```
Input: Graph $G = (V, E)$
$F(uv) = \{1, \ldots, \Delta\}, \forall uv \in E$
$Z \leftarrow \emptyset$
for $u \in V - Z$ do
    $Z \leftarrow Z \cup \{u\}$
    $W \leftarrow \{v : uv \in E\}$
    while $|W| > 0$ do
        $F \leftarrow \bigcup_{v \in W} F(uv)$
        if $\exists \alpha$ s.t. $\alpha$ appears in at most 1 set $F(uv)$ with $|F(uv)| \leq 2$, and $v \in W$ then
            Let $|F(uv)|$ be the smallest set containing $\alpha$ amongst $\{F(uw) : w \in W\}$
            $\varphi(uv) \leftarrow \alpha$
            $W \leftarrow W \setminus \{v\}$
        else
            if $|Free(u) \setminus F| = 0$ or $|F| = 0$ then
                Add $\Delta + 1$ to every $F(xy), xy \in E$
                $Z \leftarrow Z \setminus \{u\}$
                Restart colouring of $u$
            Let $\beta \in Free(u) \setminus F$
            Let $v$ be such that $|F(uv)|$ is minimal amongst $\{F(uw) : w \in W\}$
            Let $\alpha \in F(uv)$
            Exchange colours along $\beta\alpha$-path in $G[Z]$ starting at $v$
            $\varphi(uv) = \beta$
            $W \leftarrow W \setminus \{v\}$
return $\varphi$
```

Algorithm 4: BinomialGraph(n, p)

```
$G = (V, E) \leftarrow (\{1, 2, \ldots, n\}, \emptyset)$
for $u, v \in V^2$ with $v > u$ do
    if $rand() \leq p$ then
        $E \leftarrow E \cup \{uv\}$
return $G$
```

with budget b_v, we connect it to a number of vertices c_v in the core, and try to connect $b_v - c_v$ non-core vertices to v. In general, these operations may fail – for instance, we might not be able to find c_v core vertices. In that case we allow the algorithm to try to generate another graph by repeating the process up to a certain number of times. We assume we have access to a function $rand(a, b)$ that returns a uniformly sampled random integer in the range $[a, b]$. The pseudocode is given in Algorithm 5. O_k denotes the null graph on k vertices, i.e. just k vertices with no edges.

We find that most graphs generated by ExtendCore or BinomialGraph are Class 1. Hence we had to resort to using the `geng` tool from `nauty` to enumerate all graphs of small sizes, and perform experiments on graphs which have the desired properties. The ExtendCore algorithm is still useful for finding bad cores, since enumerating *all* graphs of size $n \geq 13$ is not practical

Algorithm 5: ExtendCore(H, n, Δ)

```
$m \leftarrow |V(H)|$
while we don't run out of attempts do
    $k \leftarrow rand(m+1, n)$
    $G \leftarrow H + O_{n-m}$
    $b_u \leftarrow \Delta - d_G(u), \forall u \in V(H)$
    $b_u \leftarrow rand(1, \Delta - 1), \forall u \in V(G) \setminus V(H)$
    for $u \in V(G)$ do
        $c \leftarrow 0$ if $u \in V(H)$ else $rand(1, \min(b_u, m))$
        if there are $c$ core vertices with $b_v > 0$ and $b_u - c$ non-core vertices with $b_v > 0$
         then
            Connect $u$ to those vertices
            Update $b$ appropriately
        else
            Try again
    return $G$
```

without some bounds on the structure of the graphs, as shown in Table 1, taken from the geng source code – measurements were taken on a Pentium III running at 550MHz, but even when taking into account the increase in processor speeds, it is still impractical.

n	**Number of Graphs**	**Time**
8	12346	0.11 secs
9	274668	1.77 secs
10	12005168	1.22 mins
11	1018997864	1.72 hours
12	165091172592	285 hours
13	50502031367952	10 years

Table 1: Time taken by geng to generate all graphs of size n

To generate degree sequences, we simply generate a random sequence of integers. We can place some structural constraints naturally on the generated graphs – for instance, to generate graphs with a core size of n we make sure that we have n maximal values in the degree sequence, and to generate underfull graphs we ensure that the degree sequence satisfies $|E| \leq \Delta\lfloor|V|/2\rfloor$. We find that when we pass some underfull random degree sequence to a graph realisation algorithm, the generated graph is most often Class 1.

B.1 Hilton-Zhao Conjecture (I1)

Since we were able to enumerate all graphs up to $n = 12$, we were able to confirm that the graphs we found were the only graphs which are Class 2. However, to verify our claim for larger values of n requires us to constrain the search space. We had two approaches: first, we can generate all core graphs with $\Delta = 2$ with size up to $n - 1$ containing a cycle, and then generate graphs around each of those cores using ExtendCore.

Secondly, we looked at the structure of the graphs we found, and we found that all graphs have a minimum vertex degree of $d(u) = \Delta - 1$; generating all graphs with minimum degree $\Delta - 1$ is more feasible than generating all graphs for a given size n. We used both approaches to try to find more graphs on larger values of n, but for $n >= 30$ we just used the second approach – realizing random degree sequences with a graph realization algorithm.

B.2 Bad Cores (I2)

When investigating the Bad Cores problem, we initially used Hamilton to enumerate all graphs from $n = 2$ to $n = 12$, and we only keep those which are classified as Class 2 by 1000 executions of the Vizing Heuristic (`xc`). For larger $13 \leq n \leq 30$, we used the `genrang` tool from `nauty` to generate random binomial graphs from $\mathcal{G}(n, p)$ for $p \in [0.25, 0.5, 0.75, 0.85]$. We then extracted the cores from each of the graphs and identified families of them, first by manual inspection and by writing some ad-hoc scripts using `nauty`'s isomoprhism algorithm to check if the family exists in our 'database'.

For instance, we identified that the family K_{2m}, where $m \geq 2$ is a bad core, and then checked that K_4, K_6, K_8 and K_{10} were in our database of cores. Upon identifying a potential family, we would then use the ExtendCore algorithm to try to verify our claim for larger cores.

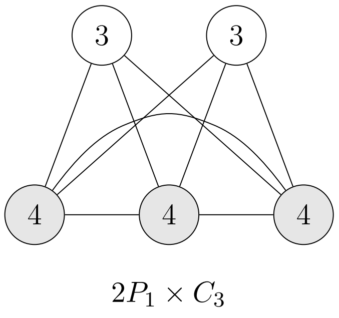
C Graph Serialisation

For our Go and Python implementations, we read and save the input graphs in JSON format, which is simple to parse but takes up relatively large amount of space. To process graphs generated by `nauty`, which serialises simple graphs in a compact ASCII format known as graph6 (*Graph Formats* 2015), we had to write a graph6 parser for both our C and Go implementations. To run our algorithms on most benchmark graphs, we also had to write a parser for the (simpler) DIMACS graph format (*DIMACS Graph Format* n.d.).

IV RESULTS

I1 – Hilton-Zhao Conjecture

We note that for $\Delta = 4$, the only connected graph which is Class 2 is $K_5 - e$, or $2P_1 \times C_3$ (Cranston & Rabern 2017). For the case of $\Delta = 5$, we found that the only connected Class 2 graph is $3P_1 \times C_4$, which is very structurally similar to that of $\Delta = 4$. As stated before, we performed exhaustive search for $n \leq 12$, but for $n > 12$ our findings remains to be proven (since we only used random testing up to $n = 200$). These two graphs are presented in Figure 6.



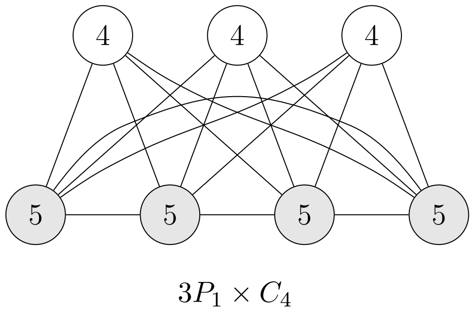


Figure 6: Only connected graphs for $\Delta = 4$ and $\Delta = 5$ (conjectured).

For other values of Δ, we found that the 'only' (same caveat as our result for $\Delta = 5$) connected Class 2 graphs possess very similar structural properties; they are presented in Table 2. We note that graphs which are Class 2 in the successive value of Δ inherit substructures from those with $\Delta - 1$; for instance, $4P_1 \times C_5 \rightarrow 5P_1 \times C_6$. We observe that in all graphs that we have found, the minimum degree of any vertex is $\Delta - 1$, and all have an odd number of vertices.

Δ			
6	$4P_1 \times C_5$	$C_4 \times C_3$	
7	$5P_1 \times C_6$ $5P_1 \times (C_3 + C_3)$	$C_5 \times C_4$	
8	$6P_1 \times C_7$ $6P_1 \times (C_3 + C_4)$	$C_6 \times C_5$ $(C_3 + C_3) \times C_5$	$C_3 \times H$
9	$7P_1 \times C_8$ $7P_1 \times (C_4 + C_4)$ $7P_1 \times (C_3 + C_5)$	$C_7 \times C_6$ $(C_3 + C_4) \times C_6$ $C_7 \times (C_3 + C_3)$ $(C_3 + C_4) \times (C_3 + C_3)$	$C_4 \times G_1$ $C_4 \times G_2$

Table 2: Class 2 connected graphs for various Δ. G_1, G_2 are the only 2 connected 4-regular graphs with 7 vertices. Note that H is the only connected 4-regular graph with 6 vertices.

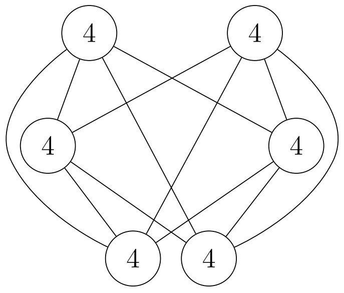


Figure 7: H graph in Table 2

I2 – Bad Cores Problem

Recall that a *Bad Core* is a graph H, such that there exists some graph G satisfying $|E(G)| \leq \Delta\lfloor |V|/2 \rfloor$ (*underfull*) which that is Class 2, and has graph H as its core. One family of bad cores is K_n where $n \geq 4$ and n is even; we managed to experimentally find vertex and edge-critical extensions of size 4 to the core, such that the resultant graph is underfull and Class 2. An extension is vertex (edge)-critical if the removal of any vertex (edge) makes the graph Class 1 or violates some conditions, e.g. K_n is no longer the core, or the graph becomes overfull. The extension graphs, as well as the cores for K_n for an even n is given in Figures 8 and 9. Note that the vertex labels are the degrees of each vertex – in Figure 9, the blue vertex with degree $n - 1$ is connected to all vertices in the core except for one connected to the red vertex.

In order to verify that our extensions were both vertex and edge-critical, we considered all possible edge and vertex deletions on our extension graphs. We found that given some deletion, either we have a Class 1 graph or we do not satisfy the preqrequisites that the graph must have K_n (even n) as a core, or the new graph is overfull.

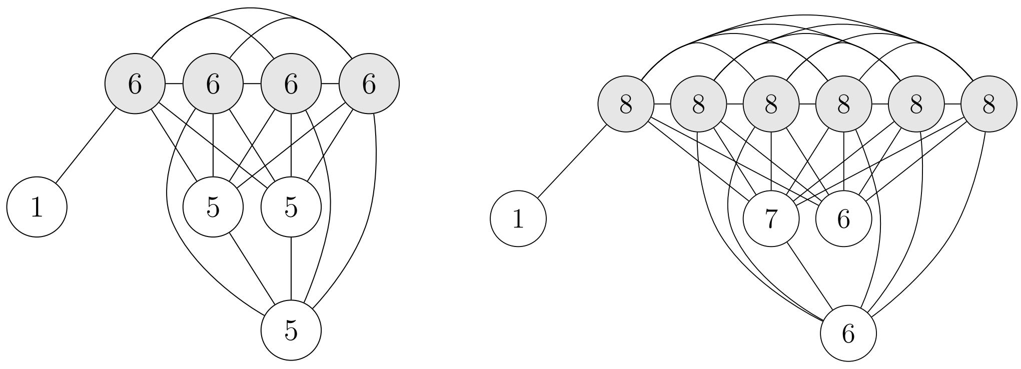


Figure 8: Extensions and cores (in grey) for K_4 and K_6, respectively.

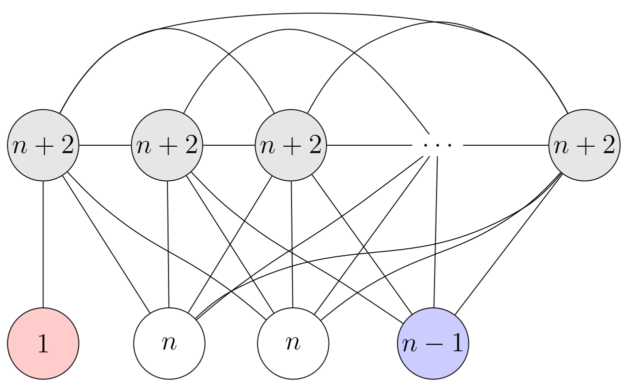


Figure 9: Extension and core for K_n (even $n \geq 8$)

We also managed to find that K_n where $n \geq 5$ is odd is a bad core, but we were not able to find a similar minimal extension for all underfull Class 2 graphs with odd complete cores. However we were able to make some observations on the degree sequences of the graphs; for K_5, the degree sequence is $(1, (n+1)^4, (n+2)^5)$, and for $n \geq 7$, it is $(1, n^2, (n+1)^2, (n+2)^n)$. Another family of bad cores we identified is $(n-1)P_1 \times C_n$ for $n \geq 3$; however we were similarly unable to identify any common substructures.

B3 – Misra-Gries vs Vizing Heuristic Benchmark

Graph instances were taken from the DIMACS graph database, and are same as the ones found by Januario (2012). We were able to replicate the performance claims; our results are given in Table 3a. The Δ column contains the maximum degree of the graphs, **MG** the number of colours used by the Misra-Gries Heuristic, **ΔVh** for the Vizing Heuristic, and **CB** for the modified Counting-based Heuristic. **Slowdown** contains the time taken by the modified Counting-Based Heuristic divided by that of the Vizing Heuristic (measurements were taken with 50 repeats).

Benchmark	**Δ**	**ΔVh**	**MG**	**CB**	**Slowdown**
myciel3	5	5	**6**	5	1.56
myciel4	11	11	**12**	11	1.59
myciel5	23	23	23	23	7.28
myciel6	47	47	47	47	4.48
myciel7	95	95	95	95	6.51
le450_5a	42	42	42	42	4.16
le450_5b	42	42	43	42	3.94
le450_5c	66	66	**67**	66	6.43
le450_5d	68	68	**69**	68	5.97
le450_15a	99	99	99	99	5.93
le450_15b	94	94	94	94	6.32
le450_15c	139	139	**140**	139	10.18
le450_15d	138	138	138	138	8.49
le450_25a	128	128	128	128	5.83
le450_25b	111	111	111	111	6.42
le450_25c	179	179	179	179	10.68
le450_25d	157	157	157	157	9.80

(a) Small benchmark graphs

Benchmark	**Δ**	**ΔVh**	**LH**
4-FullIns_4	119	119	119
5-FullIns_4	160	160	160
ash331GPIA	23	23	23
ash958GPIA	24	24	24
qg.order30	58	**59**	58
qg.order60	118	**119**	118
wap04a	351	351	351
*DSJC500.1**	68	68	69
*will199GPIA**	38	38	40
qg.order100	198	**199**	212
latin_square_10	683	**684**	-

(b) Large benchmark graphs

Table 3: Results for various benchmark graphs

A3 – Large(r) Benchmark Graphs

Our results in Table 3b for large $n > 500$ benchmark graphs are divided into two sections – graphs with and without a known edge-chromatic index (Hilgemeier et al. 2003); the latter is given with *italicized* names. The **LH** column contains the best result found by heuristics in (Hilgemeier et al. 2003), and the **ΔVh** column is the best result found by `lxc`. In only 4 cases (in **bold**) we were not able to find an edge-colouring using only Δ colours, and in 2 cases our implementation performed worse than that of Hilgemeier et al. (2003). We found new results (they are Class 1 and no previous better results were found to the best of our knowledge) for the benchmark graphs marked with '*'. Comparisons with the latin_square_10 graph was omitted since our graph properties differed from that in the paper.

B4 – Empirical Confirmation of Erdös & Wilson's results on $\mathcal{G}(n, p)$

Figure 10 shows our results on $\mathcal{G}(n, p)$ for various values of p. For any value of $p < 1$, we find that the probability that some graph $G \sim \mathcal{G}(n, p)$ is Class 1 converges to 1 rapidly. For $p = 0.95$, the probability fluctuates (but still converges); graphs from $\mathcal{G}(n, 0.95)$ are very similar (up to a certain n) to complete graphs K_n. We calculated the probabilities using our Go implementation of the Vizing Heuristic on 500 samples for each value of n.

I4 – Vizing Heuristic vs Counting-Based Heuristic

Figure 11 shows the performance of the unmodified Counting-Based Heuristic (CB), modified Counting-Based Heuristic (CB*), and the Vizing Heuristic (VH) on $\mathcal{G}(n, p)$. Without our

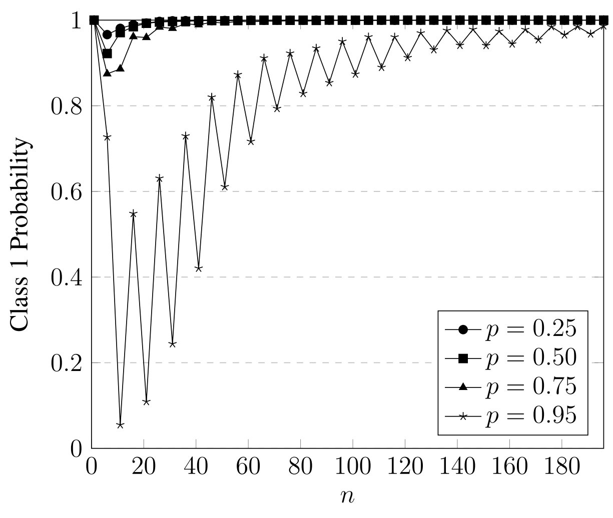


Figure 10: $\mathcal{G}(n, p)$ Class 1 Probabilities for various values of p

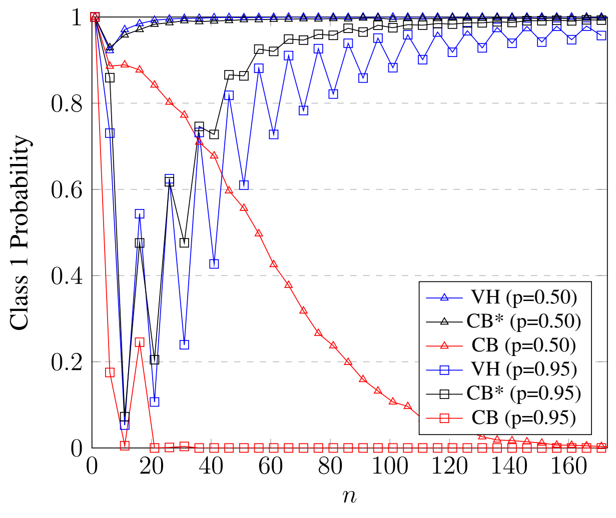


Figure 11: Modified and unmodified Counting-Based vs Vizing Heuristic on $\mathcal{G}(n, p)$

modification, the Counting-Based Heuristic performs much worse than the Vizing Heuristic when n increases. Our modification is able to improve performance considerably and perform competitively ($p = 0.50$), or even better than ($p = 0.95$) the Vizing Heuristic.

V EVALUATION

A *Conjectures*

We present a number of conjectures for cases in the Hilton-Zhao Conjecture and the Bad Cores problem stemming from our computational experiments. Due to the lack of time and expertise we were not able to formally prove any of them, but they perhaps serve as a good exploratory study and starting point for any future work. For Conjecture 2, because $3P_1 \times C_4$ is so structurally similar to $2P_1 \times C_3$, it is most likely to be true.

Conjecture 2 *$3P_1 \times C_4$ is the only connected graph that is Class 2 for the case $\Delta = 5$ of the Hilton-Zhao Conjecture.*

Conjecture 3 *The graphs given in Table 2 are the only connected graphs that are Class 2 for their respective values of Δ satisfying the Hilton-Zhao Conjecture.*

Conjecture 4 *Graphs which satisfy the Hilton-Zhao conjecture with maximum degree Δ have minimum degree $\Delta - 1$, and have an odd number of vertices.*

We conjecture that the set of extensions in Figures 8 and 9 is minimal, in that they are the smallest extensions in terms of the number of edges and vertices to any K_n (with an even $n \geq 4$). Should Conjecture 5 be proven, it would imply that any graph with a complete even core and a semicore containing our extension graphs as an induced subgraph is Class 2.

Conjecture 5 *The extensions given in Figures 8 and 9 are the minimal extensions for an even-complete core of size ≥ 4 (K_n for all even $n \geq 4$).*

Conjecture 6 *K_n (for all odd $n \geq 5$) is a bad core, with a minimal extension of size 5 and degree sequence $(1, 6^4, 7^5)$ if $n = 5$, and $(1, n^2, (n+1)^2, (n+2)^n)$ otherwise.*

Conjecture 7 *$(n-1)P_1 \times C_n$ is a bad core, for all $n \geq 3$.*

B Implementations

We were able to write correct and good implementations of the edge-colouring heuristics that we have considered, as well as fully replicate the results claimed by Januario, T. (2012), and we were also able to make improvements on the Counting-Based Heuristic based on ideas introduced in the same paper, to provide better colouring performance in practice. Our implementations frequently find the optimal edge colouring for most benchmark graphs.

Our C implementation of the Vizing Heuristic was also able to find new results for some large benchmark graphs (see Table 3b), and is able to robustly handle large graphs with 10000 vertices (e.g. the qg.order100 benchmark graph) with no issues. Our implementation also has a much smaller runtime compared to the ant1 heuristic by Hilgemier et al. (2003) – on the qg.order100 graph, we compute a $\Delta + 1$ edge-colouring in only 21.3 seconds, whilst the ant1 heuristic took 248.38 seconds and found an edge-colouring using $212 > \Delta + 1$ colours. Even taking into account that they were using an 1.8 GHz machine and our performance measurements were taken on a 3.4 GHz machine, the decrease in running time is still significant.

One of the primary advantages of using Hamilton was that we were able to perform more and larger experiments. This led to us improving the confidence we have in our results as well as being able to find our results in a reasonable amount of time.

VI CONCLUSIONS

In this project we were able to computationally investigate the Hilton-Zhao conjecture as well as the bad-cores problem. Since Class 2 graphs are very rare in practice, we had to restrict our search to special cases in order to make any progress. Our work may be used in the future by heuristics that estimate whether a graph is Class 1 or 2 – by checking if it contains some induced subgraph and core that is contained in one of our conjectures, we can say if a graph is Class 1 or 2 with some chance of error.

One possible extension to our work is to consider parallelized algorithms for edge-colouring, for instance those introduced by Karloff Shmoys (1987). Since most of our graphs were quite small in our experiments, we were able to perform experiments quickly with just the serial Vizing Heuristic, but to test larger graphs we might have to use parallel algorithms.

On a more theoretical note, another extension of our work is to consider other graph properties; for instance, the treewidth (Bruhn et al. 2015) and how it relates to the edge chromatic number χ'. Yet another possible extension is to consider the probability of a graph being Class 2 when it is from different graph distributions – for instance, it might be quite interesting to observe how, in the Barabási-Albert Model, does the probability of a graph being Class 2 from a distribution parameterised by n, m_0, and m change depending on the values of those parameters – we performed some preliminary analysis given in Figure 12.

It might also be interesting and helpful to investigate Class 2 probabilities in graph models where certain graph structures are generated by construction – for instance, the classification of split graphs is still an open research problem, but it might be easier to investigate the probabilistic

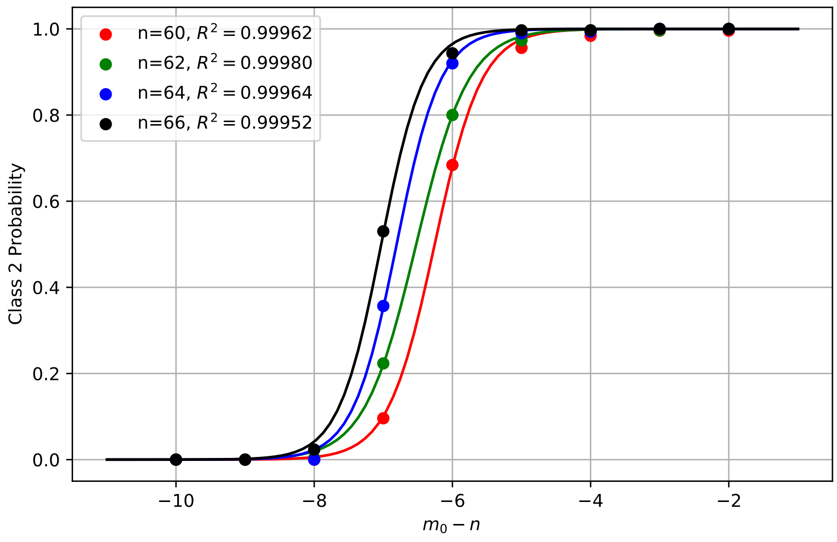


Figure 12: Class 2 probabilities for the Barabási-Albert Model for various values of $m = m_0$ and n. For each value of n, we were able to fit a sigmoid curve $1/(1 + e^{-k(x-x_0)})$ parameterized by k and x_0. We also give the R^2 regression values between our sigmoid fit and the real data.

model defined as follows. We say $G = (V, E)$ is a split graph (Tyškevic & Cernjak 1979) if it's V can be partitioned into two sets $|V_1| = n$, and $|V_2| = r$, such that $G[V_1] = K_n$, and $G[V_2] = O_r$ (i.e. a complete graph of size n, and an independent set with r vertices). One can define a new graph model as follows: $S(n, r, p)$, where n is the size of the complete graph, and r the size of the independent set, and p the probability of an edge existing between a vertex of the complete graph and a vertex of the independent set.

References

Barabási, A.-L. & Albert, R. (1999), 'Emergence of scaling in random networks', *Science* **286**(5439), 509–512.

Bayati, M., Kim, J. H. & Saberi, A. (2010), 'A sequential algorithm for generating random graphs', *Algorithmica* **58**(4), 860–910.

Behzad, M., Chartrand, G. & Cooper, J. K. (1967), 'The colour numbers of complete graphs', *Journal of the London Mathematical Society* **s1-42**(1), 226–228.

Bruhn, H., Lang, R. & Stein, M. (2015), 'List edge-coloring and total coloring in graphs of low treewidth', *Journal of Graph Theory* **81**(3), 272–282.

Cao, Y., Chen, G., Jing, G., Stiebitz, M. & Toft, B. (2019), 'Graph edge coloring: A survey', *Graphs and Combinatorics* **35**(1), 33–66.

Chetwynd, A. & Hilton, A. (1990), 'The chromatic index of graphs with large maximum degree, where the number of vertices of maximum degree is relatively small', *Journal of Combinatorial Theory, Series B* **48**(1), 45–66.

Cranston, D. W. & Rabern, L. (2017), 'The hilton–zhao conjecture is true for graphs with maximum degree 4'.

DIMACS Graph Format (n.d.), http://lcs.ios.ac.cn/c̃aisw/Resource/about_DIMACS_graph_format.txt. Accessed: 2020-03-06.

Ehrenfeucht, A., Faber, V. & Kierstead, H. (1984), 'A new method of proving theorems on chromatic index', *Discrete Mathematics* **52**(2-3), 159–164.

Enochs, B. P. & Wainwright, R. L. (2001), "forging" optimal solutions to the edge-coloring problem, *in* 'Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation', GECCO'01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 313–319.

Erdős, P. & Wilson, R. J. (1977), 'On the chromatic index of almost all graphs', *Journal of Combinatorial Theory, Series B* **23**(2-3), 255–257.

Erdös, P. et al. (1959), 'On random graphs'.

Fournier, J. & JC, F. (1977), 'Methode et theoreme general de coloration des aretes d'un multigraphe.'.

Galby, E., Lima, P. T., Paulusma, D. & Ries, B. (2019), 'On the parameterized complexity of k-edge colouring'.

Gould, R. (1988), *Graph theory*, Benjamin/Cummings Pub. Co.

Graph Formats (2015), http://users.cecs.anu.edu.au/b̃dm/data/formats.txt. Accessed: 2020-03-06.

Hakimi, S. L. (1962), 'On realizability of a set of integers as degrees of the vertices of a linear graph. i', *Journal of the Society for Industrial and Applied Mathematics* **10**(3), 496–506.

Hilgemeier, M., Drechsler, N. & Drechsler, R. (2003), Fast heuristics for the edge coloring of large graphs, *in* 'Euromicro Symposium on Digital System Design, 2003. Proceedings.', IEEE.

Holyer, I. (1981), 'The NP-completeness of edge-coloring', *SIAM Journal on Computing* **10**(4), 718–720.

Januario, T. (2012), An edge coloring heuristic based on vizing's theorem.

Karloff, H. J. & Shmoys, D. B. (1987), 'Efficient parallel algorithms for edge coloring problems', *Journal of Algorithms* **8**(1), 39–52.

Khuri, S., Walters, T. & Sugono, Y. (2000), A grouping genetic algorithm for coloring the edges of graphs, *in* 'Proceedings of the 2000 ACM symposium on Applied computing - SAC'00', ACM Press.

Machado, R. C. & de Figueiredo, C. M. (2010), 'Decompositions for edge-coloring join graphs and cobipartite graphs', *Discrete Applied Mathematics* **158**(12), 1336–1342.

McKay, B. D. & Piperno, A. (2014), 'Practical graph isomorphism, II', *Journal of Symbolic Computation* **60**, 94–112.

Misra, J. & Gries, D. (1992), 'A constructive proof of vizing's theorem', *Information Processing Letters* **41**(3), 131–133.

Skiena, S. S. (2012), 16.8 edge colouring, *in* 'The Algorithm Design Manual', Springer London, pp. 548–550.

Tyškevic, R. I. & Cernjak, A. (1979), 'Canonical decomposition of a graph determined by the degrees of its vertices. vestsı akad. navuk bssr ser. fız', *Mat. Navuk* **5**(5), 14–26.

Vizing, V. G. (1964), On an estimate of the chromatic class of a p-graph.

Vizing, V. G. (1965), 'Critical graphs with given chromatic class (in russian)', *Metody Discret. Analiz.* **5**, 9–17.