

Article

Efficient Direct Reconstruction of Bipartite (Multi)Graphs from Their Line Graphs Through a Characterization of Their Edges

Drago Bokal ^{1,2,3}  and Janja Jerebic ^{4,*} ¹ Faculty of Natural Sciences and Mathematics, University of Maribor, 2000 Maribor, Slovenia; drago.bokal@um.si² Institute of Mathematics, Physics and Mechanics, 1000 Ljubljana, Slovenia³ DataBitLab d.o.o., Perhavlčeva Ulica 19, 2000 Maribor, Slovenia⁴ Faculty of Organizational Sciences, University of Maribor, 4000 Kranj, Slovenia

* Correspondence: janja.jerebic@um.si

Abstract

We study the line graphs of bipartite multigraphs, which naturally arise in combinatorics, game theory, and applications such as scheduling and motion planning. We introduce a new characterization of these graphs via valid partial assignments of the edges of the underlying bipartite multigraph to the vertices of its line graph. We show that an empty assignment extends to a complete one precisely when the graph is a line graph of a bipartite multigraph. Based on this, we design an $O(\Delta(G) |E(G)|)$ algorithm that incrementally constructs such assignments. The algorithm also provides a data structure supporting efficient solutions to problems of maximum clique, maximum weighted clique, minimum clique cover, chromatic number, and independence number. For line graphs of bipartite simple graphs these problems become solvable in linear time, improving on previously known polynomial-time results. For general bipartite multigraphs, our method enhances the $O(|V(G)|^3)$ recognition algorithm of Peterson and builds on the results of Demaine et al., Hedetniemi, Cook et al., and Gurvich and Temkin.

Keywords: UNO-graph; line graph; bipartite graph; bipartite multigraph; graph algorithm**MSC:** 05C57; 05C69; 05C76; 05C85; 68R10

Academic Editor: Randy R. Davila

Received: 17 July 2025

Revised: 22 August 2025

Accepted: 3 September 2025

Published: 5 September 2025

Citation: Bokal, D.; Jerebic, J. Efficient Direct Reconstruction of Bipartite (Multi)Graphs from Their Line Graphs Through a Characterization of Their Edges. *Mathematics* **2025**, *13*, 2876. <https://doi.org/10.3390/math13172876>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Line graphs of bipartite (multi)graphs have naturally appeared in several contexts. For our treatise, the two most relevant previous expositions are those of Demaine et al. [1], who treat them as a model of compatible UNO game cards, and of Peterson [2], who interprets them as blow-ups of gridline graphs. There, p D gridline graphs are defined as the graphs isomorphic to a graph whose vertices are distinct points in $\mathbb{R}^p$, any two being adjacent if and only if the corresponding points differ in exactly one coordinate. The blow-ups of these graphs are defined analogously, except that distinctness of points is not required. UNO graphs, on the other hand, are defined by Demaine et al. as isomorphic to graphs whose vertices are pairs (or tuples for p -dimensional versions) of values of properties, two vertices being adjacent if they share a value of at least one property. Again, we obtain simple graphs if and only if the pairs are distinct.

For $p = 2$, the two concepts coincide: A graph G is a 2D gridline graph G or a 2D UNO graph if its vertices can be assigned two coordinates/properties, such that two vertices are adjacent if and only if they share precisely one coordinate/property. This corresponds

to G being a line graph of a bipartite (multi)graph H , with the values of each property corresponding to one partition set of the vertices of H , and the vertices of G corresponding to the edges of H . Note that p -dimensional gridline graphs no longer correspond to line graphs of p -partite graphs, whereas p -dimensional UNO graphs correspond to line graphs of p -partite hypergraphs in which each hyperedge is incident with precisely one vertex of each partition of the vertices. Given that our algorithms build upon graphs being line graphs of bipartite multigraphs, in our paper we denote the graphs of interest as UNO graphs. Thus, in addition to emphasizing the inspiring analogy, we also emphasize the playful aspect of the topic, possibly opening up new venues of research and exposing it to a younger audience.

In Demaine et al. [1], UNO p graphs were studied from the viewpoint of algorithmic combinatorial game theory. The mathematical models for several versions of UNO (a card game invented by Merle Robbins in 1971) were defined, and their computational complexities were analyzed based on the close connection of the game with the Edge Hamiltonian Path Problem. In particular, they observed that a single-player version is equivalent to finding a Hamiltonian path in the corresponding UNO-1 (or simply UNO) graph, which was shown to be NP-complete. The parameterized complexity of the Edge Hamiltonian Path Problem was studied in [3,4]. UNO graphs, however, appeared earlier under different names. Our results also build upon the earlier independent work of the following: Hedetniemi [5], who referred to them as graphs of (0,1)-matrices; Cook et al. [6], who named them adjacency graphs; and Gurvich and Temkin [7], where the English translation referred to them as checked graphs.

In line with the earlier argument, Demaine et al. [1] observed that a graph G is an UNO graph if and only if G is the line graph of a bipartite graph. Much earlier, in [5], gridline graphs were also characterized as line graphs of bipartite graphs. Moreover, it is known that G is a line graph of a bipartite graph if and only if G contains no induced claw, diamond, or odd hole [5,8,9]. Another equivalent characterization, given in [2], states that G is a line graph of a bipartite graph if and only if G is diamond-free and the graph of maximal cliques of G is bipartite.

Roussopoulos [10] devised a linear-time algorithm to verify whether G is a line graph of some graph H . His algorithm also produces the graph H . It is folklore that the bipartiteness of a graph can be checked in linear time; hence, these two algorithms together yield a linear-time verification of whether a given graph G is a line graph of a bipartite simple graph (and, thus, it satisfies any of the equivalent conditions listed in Theorem 1).

Theorem 1 ([1,2,5,8,9]). *The following are equivalent:*

- (i) G is an UNO graph.
- (ii) G is a line graph of a bipartite graph.
- (iii) G is a two-dimensional gridline graph.
- (iv) G contains no induced claw, diamond, or odd hole.
- (v) G is diamond-free, and the graph of the maximal cliques of G is bipartite.

In this paper, we define and characterize k -UNO graphs as line graphs of bipartite multigraphs with edge multiplicity at most k . Our main contributions are twofold: on the structural side, we introduce a novel edge classification into forced, semi-constrained, and unconstrained edges, and we show that valid partial assignments extend to a complete one if and only if the graph is a line graph of a bipartite multigraph. On the algorithmic side, we exploit this characterization to design a direct $O(\Delta(G)|E(G)|)$ time recognition algorithm. This improves upon the $O(|V(G)|^3)$ algorithm of Peterson for multigraphs, while reducing to linear time $O(|E(G)|)$ for simple line graphs of bipartite graphs. Moreover, the data structure underlying the algorithm provides efficient solutions to several fundamental

problems on line graphs of bipartite (multi)graphs, including maximum clique, maximum weighted clique, minimum clique cover, chromatic number, and independence number. Since line graphs of bipartite (multi)graphs are perfect [11–13], these problems are solvable in linear time for line graphs of bipartite simple graphs—improving previously known polynomial-time results [14–16]. Taken together, our results establish a unified framework that advances both the structural understanding and the algorithmic tractability of this important graph class.

We conclude the introduction by pointing towards several applications of k -UNO graphs in diverse areas of mathematics and applied sciences. In combinatorics, they play a central role in proving the Dinitz conjecture via list edge coloring of bipartite graphs [17]. In matching theory and economics, kernels in such graphs correspond to stable marriage [12]. In scheduling and timetabling, vertex colorings of gridline graphs model time slot assignments [18]. In enumerative combinatorics, rook polynomials that are used to count non-attacking configurations are also studied in Roberts’ foundational work [18] and have been recently revisited in the context of grid polyominoes [19]. Finally, gridline graphs support motion planning in robotics in grid-like environments where minimizing turns is critical [20]. To make these applications more concrete, we illustrate two representative cases where k -UNO graphs naturally arise. In scheduling and timetabling, time slot assignments can be modeled by vertex colorings of gridline graphs, where adjacency encodes conflicts between tasks [18]. In robotics and motion planning, gridline graphs describe the feasible moves of a robot navigating a grid-like environment, with UNO assignments reflecting row/column positions and movement orientations [2,20]. These examples demonstrate how the structural properties studied in this paper translate directly into real-world constraints, thereby underlining the practical relevance of our theoretical results.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts and notation used throughout the paper. Section 3 presents the structural foundation of k -UNO graphs. We define the edge classification (forced, semi-constrained, unconstrained), prove a complete characterization in terms of clique intersections and the bipartite nature of the clique graph, and introduce several lemmas that reveal local structural properties crucial for algorithmic applications. Section 4 develops the recognition algorithm, based on incremental construction of valid partial assignments. We conclude with an overview and related open problems in Section 5.

2. Preliminaries and Notation

For a graph G , we denote by $V(G)$ the vertex set of G and by $E(G)$ the edge set of G . A graph $G = G(V(G), E(G))$ is connected if for every pair of vertices $u, v \in V(G)$ there exists a path in G from u to v . Since the k -UNO graph problem considered in this paper naturally reduces to an analysis on connected components, we restrict our attention to connected graphs. For a subset of vertices $S \subseteq V(G)$, the induced subgraph $G[S]$ is the graph whose vertex set is S , where two vertices u and v are adjacent in $G[S]$ if and only if they are adjacent in G . A *clique* in a graph G is a vertex subset of $V(G)$ that induces a complete subgraph of G . The *clique number* $\omega(G)$ of G is the largest order of a clique in G . A clique of G is *maximal* if it is not contained in any other clique of G . The *clique graph* $M(G)$ is the intersection graph of the maximal cliques of G (the vertices of $M(G)$ are the maximal cliques of G , and two vertices of $M(G)$ are adjacent if and only if the corresponding maximal cliques share a vertex in G). G is a *bipartite* graph if its vertex set can be partitioned into two disjoint subsets $V(G) = A + B$, such that every edge of G connects a vertex in A to a vertex in B (for a bipartite multigraph, multiple edges are allowed). G is a *complete bipartite graph* or a *bi-clique* with bipartition $V(G) = A + B$ if and only if G is a bipartite graph for which each vertex from A is adjacent to each vertex in B .

For $|A| = m$ and $|B| = n$, we denote such a graph by $K_{m,n}$. For example, the graph $K_{1,3}$ is called a *claw*. A graph is *claw-free* if and only if it does not contain $K_{1,3}$ as an induced subgraph. The *line graph* $L(G)$ of a graph G is a graph with $V(L(G)) = E(G)$, in which two vertices e and f of $L(G)$ are adjacent if and only if e and f are incident edges in G . A p -dimensional *gridline graph*, where $p \in \mathbb{N}$, is a graph G that is isomorphic to some graph $\overline{G}$ (called a realization of G) whose vertices are located in $\mathbb{R}^p$, no two vertices have the same coordinates, and vertices at $x = (x_1, \dots, x_p)$ and $x' = (x'_1, \dots, x'_p)$ are adjacent whenever they differ in exactly one entry. For further standard definitions and background, we refer the reader to [21].

Hereinafter, we define a k -UNO graph for a given multiset of UNO cards (in games where multiple identical cards are allowed).

Definition 1. Let $A = \{a_1, \dots, a_m\}$ be a set of numbers, and let $B = \{b_1, \dots, b_n\}$ be a set of colors. Let $C \subset A \times B \times \mathbb{Z}_{k+1}$ be a multiset of UNO cards, such that each combination of a number and a color can appear at most k times. $G(C)$ is a k -UNO graph for the multiset C if the vertices of G are elements of C , and two vertices are adjacent if and only if they share either the same number, the same color, or both. Let $\kappa(C)$ denote the largest multiplicity of a card in C .

An abstract graph G is a k -UNO graph if there exists a multiset C of multiplicities at most k , such that G is isomorphic to the k -UNO graph $G(C)$. The isomorphism that assigns cards of C to vertices of G is called an UNO assignment.

Note that each UNO-1 graph defined by Demaine et al. for a single player in [1] is precisely a 1-UNO graph for a single player (we refer to these graphs simply as UNO graphs). Also, each line graph of a bipartite multigraph with edge multiplicity at most k can be interpreted as a k -UNO graph. Every multigraph considered in this paper is bipartite and, thus, has no loops.

To make the definition of an UNO assignment more concrete, we illustrate it with a small example. Consider the set of numbers $A = \{1, 2, 3\}$ and the set of colors $B = \{\text{red}, \text{blue}\}$. The corresponding multiset of cards is

$$C = \{(1, \text{red}), (2, \text{red}), (3, \text{red}), (1, \text{blue}), (2, \text{blue}), (3, \text{blue})\}.$$

The 1-UNO graph $G(C)$ has six vertices, one for each card. Two vertices are adjacent if and only if the corresponding cards share the same number or the same color. Thus, for instance, $(1, \text{red})$ is adjacent to $(1, \text{blue})$ because they share the number 1, and it is also adjacent to $(2, \text{red})$ and $(3, \text{red})$ because they share the color red. The resulting graph is shown in Figure 1, where horizontal edges represent equal numbers and vertical/curved edges represent equal colors. This example demonstrates how UNO assignment naturally induces a graph structure, and, in fact, the graph obtained here is isomorphic to the line graph of $K_{3,2}$.

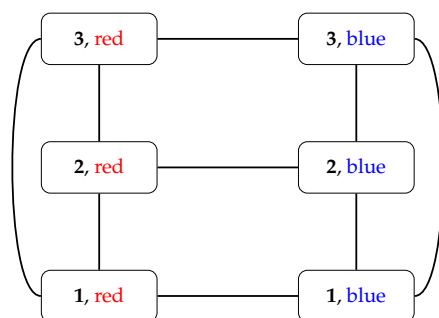


Figure 1. UNO assignment with six cards: $(1, \text{red}), (2, \text{red}), (3, \text{red}), (1, \text{blue}), (2, \text{blue}), (3, \text{blue})$. Vertices represent cards; two vertices are adjacent if they share the same number (horizontal edges) or the same color (vertical/curved edges). The resulting graph is a 1-UNO graph isomorphic to the line graph of $K_{3,2}$.

3. Characterizations

In Demaine et al. [1], UNO graphs were identified as an interesting direction of research because they form a subclass of claw-free graphs and appear to have interesting properties of their own. Our main structural result is the characterization of the k -UNO graphs given in Theorem 3. This result can be seen as an extension of Peterson’s characterization of line graphs of bipartite multigraphs (see Theorem 2):

Theorem 2 (Peterson [2], Theorem 4.1). *The following statements are equivalent:*

- (i) G is a line graph of a bipartite multigraph;
- (ii) $M(G)$ is bipartite.

In our case, we consider multigraphs with a given maximum edge multiplicity. We begin with a useful observation (Lemma 1) about cliques in k -UNO graphs.

Lemma 1. *Let G be a k -UNO graph and K a clique in G . Then, either all the vertices of K have the same color or all share the same number.*

Proof. If all vertices of K have the same color, we have finished. Suppose two vertices u, v of K have different colors. As they are adjacent in K , u and v must share the same number. Let w be any other vertex of K , implying that w is adjacent to both u and v . If w does not have the same number as u and v then it must have the same color as both of them—a contradiction. Thus, w has the same number as u and v . $\square$

Theorem 3. *The following are equivalent:*

- (i) G is a line graph of a bipartite multigraph with edge multiplicity at most k ;
- (ii) G is a k -UNO graph;
- (iii) G is claw-free, any two maximal cliques of G share at most k vertices, and $M(G)$ is bipartite;
- (iv) Any two maximal cliques of G share at most k vertices, and $M(G)$ is bipartite.

Proof. ($i \Rightarrow ii$): Suppose G is the line graph of a bipartite multigraph $H = (A + B, C)$ with edge multiplicity at most k . The choice of letters is intentional: interpret A as the set of numbers and B as the set of colors. Then, each edge in C corresponds to a card labeled by a number–color pair. In $G = L(H)$, the vertices are the edges C , and two vertices are adjacent if and only if the corresponding edges in H share an endpoint—that is, they share a number or a color. Hence, G is a k -UNO graph over A, B , with multiplicities bounded by k .

($ii \Rightarrow iii$): Suppose G is a k -UNO graph. To prove claw-freeness, consider any vertex a with three neighbors b, c, d . If no two of them share the same color as a then at least two must share the same number, implying adjacency. So G is claw-free. Let K_1 and K_2 be distinct maximal cliques. By Lemma 1, if both K_1 and K_2 are number-cliques or both are color-cliques, then they are disjoint. Otherwise, suppose K_1 is a number-clique for a and K_2 is a color-clique for b . Then, the intersection $K_1 \cap K_2$ contains only cards with number a and color b , of which there are at most k . To show that $M(G)$ is bipartite, label each maximal clique with α (number) or β (color) according to its type; ties can be broken arbitrarily. This gives a valid two-coloring of $M(G)$, proving (iii).

($iii \Rightarrow iv$): Immediate.

($iv \Rightarrow i$): Let G satisfy (iv). Color $M(G)$ with two labels, α and β , to form bipartite sets A_1 and B_1 of maximal cliques. Let A_2 (resp. B_2) be the set of singleton sets for vertices not appearing in any clique of A_1 (resp. B_1). Each vertex belongs to exactly one element in $A = A_1 \cup A_2$ and one in $B = B_1 \cup B_2$, and any pair $a \in A, b \in B$ shares at most k vertices. Define a bipartite multigraph $H = (A + B, E)$, where ab appears with multiplicity equal to $|a \cap b|$. Then, H has edge multiplicity at most k , and $G = L(H)$. $\square$

Note that the set C can be reconstructed from its 1-UNO graph (up to a permutation of colors and numbers and exchange of the roles of colors and numbers). However, for a multiset C this is not necessarily true. If in some subset $C' \subseteq C$ of the cards of the same color all have the same number, and if no other card has that number, then there is no maximal clique corresponding to that unique number. Thus, $G(C)$ is the same as in the case where each card in C' has a different number.

In the above characterization proof, this is addressed through the singleton sets in A_2 and B_2 , which ensure that any two cards are different if the constraints of the clique structure permit. The opposite extreme would be if the sets in A_2 were to partition the vertices in no maximal clique of A_1 according to the maximal cliques of B_1 that they belonged to. Analogously, the vertices in no maximal clique of B_1 would be in the sets of B_2 according to the maximal cliques of A_1 that these vertices belonged to. Such a partition would render two cards equal whenever the clique structure of G permitted it. This discussion poses no issue for the above characterization but illustrates the loss of information when converting a bipartite multigraph to its line graph. The inverse problem may not have a unique solution, and even the maximal multiplicity of the reconstructed multigraph may differ. The issue is most rigorously understood through the following proposition:

Proposition 1. *Let G be a multigraph with vertices $v_0, v_1, \dots, v_n$ and edges v_0v_i , $i = 1, \dots, n$, where $k_i \in \mathbb{N}$ is the edge multiplicity of v_0v_i . Let $k = \sum_{i=1}^n k_i$. Then, the complete graph K_k is the line graph of G .*

Proof. The claim follows from the observation that all the edges in the defined graph share the endvertex v_0 and, therefore, correspond to adjacent vertices in the line graph. $\square$

This freedom plays a central role in our recognition algorithm. Note that when analyzing vertices for a linear algorithm only their local neighboring structure is available, and queries must be efficient. Understanding the structure of the graph around the vertices and efficient detection of deviations from that structure is, therefore, a key to linearity of the algorithm. First, we significantly limit the number of cliques a vertex of a k -UNO graph is in:

Lemma 2. *Let G be a k -UNO graph. Then, any vertex lies in at most two maximal cliques of G .*

Proof. By Theorem 3, the intersection graph $M(G)$ of maximal cliques of G is bipartite and has no triangles. Should a vertex of G lie in three maximal cliques, those three cliques would form a triangle of $M(G)$, which is a contradiction. $\square$

Next, we formalize the interesting properties of vertices and edges and characterize them in the rest of this section.

Definition 2. *Let v be a vertex of G . We say that v is unconstrained if it lies in precisely one maximal clique of G . An edge is unconstrained if both its endpoints are unconstrained. An edge is constrained if it is not unconstrained.*

Obviously, each vertex is in at least one maximal clique of G . Therefore, for a vertex to be unconstrained it suffices to show that it lies in no two maximal cliques. Also, since the endpoints of an unconstrained edge share all maximal cliques they necessarily lie in the unique shared maximal clique. This gives rise to the following characterization, which is key in proving the correctness of our recognition algorithm:

Lemma 3. *Let $e = uv$ be an edge of a graph G . Then, e is unconstrained if and only if u and v share the same set of neighbors and every pair of their neighbors is adjacent.*

Proof. Without loss of generality, suppose for contradiction that u and v are unconstrained, but u has a neighbor w not adjacent to v . Then, uw lies in a maximal clique containing u and w but not v , and uv lies in a maximal clique not containing w —a contradiction to u being unconstrained. Hence, u and v have the same set of neighbors.

Suppose now that the neighbors x, y of u and v are not adjacent. Then, xu and vy lie in distinct maximal cliques, again contradicting the assumption that u is unconstrained.

Conversely, suppose u and v share the same set of neighbors and all those neighbors are pairwise adjacent. Then, their common neighborhood forms a clique K in G . Any vertex not in K is not adjacent to u or v , since it is either not a common neighbor or not adjacent to some neighbor. Hence, K is a maximal clique containing both u and v , and no other clique can contain either. Thus, u , v , and uv are unconstrained. $\square$

In a k -UNO graph, the way that constrained edges share maximal cliques is very restricted:

Lemma 4. *Let G be a k -UNO graph and let $e = uv$ be its constrained edge:*

- (i) *e lies in precisely two maximal cliques of G , or e lies in one maximal clique but one of its endvertices lies in precisely two;*
- (ii) *if e lies in two maximal cliques then its endvertices share the same set of neighbors.*

Proof. As e is constrained, without loss of generality we assume u is constrained and lies in at least two maximal cliques of G . By Lemma 2, u lies in precisely two. Then, v either lies in both these cliques or in exactly one of them. If in both then Lemma 2 implies these are all the cliques containing u or v , and, hence, the edge uv . If in one, there may be a third clique containing u but not v . Hence, it does not contain uv , establishing Lemma 4 (i).

For Lemma 4 (ii), let e be in two maximal cliques that are shared by u and v . If there is a vertex w incident to just one of them, say u , then the edge uw is in a maximal clique not containing v . This violates the bipartiteness of $M(G)$, establishing the claim. $\square$

Lemmas 3 and 4 identify the key properties of edges for recognition of k -UNO graphs: they are either unconstrained or not, and their endpoints either share the set of neighbors or do not. Endpoints of any unconstrained edge share their set of neighbors, but for constrained edges either case is possible.

We define an edge to be *forced* if it is constrained and its endpoints share the set of neighbors. If constrained but not forced, we call the edge *semi-constrained*. This is justified by the following lemma:

Lemma 5. *Let $e = uv$ be a forced edge of a k -UNO graph G . Then, in any UNO assignment of G , u and v are assigned the same color and the same number.*

Proof. As e is forced, it lies in two distinct cliques of G by Lemma 4. By Lemma 1, the vertices of one of these cliques all share the same number, and the vertices of the other all share the same color. Therefore, u and v share both number and color. $\square$

This lemma has a natural consequence in 1-UNO graphs, which follows from the observation that any two vertices of G assigned the same number and color correspond to parallel edges in the graph whose line graph G is of. In a k -UNO graph, such vertices are endvertices of either fixed or unconstrained edges, and equality is not a requirement for the latter, whereas equal assignment is needed for the forced edges.

Lemma 6. *If G is a line graph of a bipartite simple graph then no edge of G is fixed.*

The Lemma is key to understanding that our approach yields a linear algorithm for recognizing line graphs of simple bipartite graphs, cf. Theorem 5 (a.vi).

Lemma 7. Let $e = uv$ be a semi-constrained edge of a k -UNO graph G . Then, in any UNO-assignment of G , u and v share one property but differ in the other.

Proof. Since e is constrained, one of u, v lies in two maximal cliques, and since e is not forced, one of those cliques does not contain the other vertex. Assume u is such a vertex. Let K_u be a clique containing u but not v , and let K_v be the clique containing both.

There is a property shared by all vertices of K_v , which is the property shared by u and v . On the other hand, all vertices of K_u share another property. Since v is not in K_u there exists a vertex in K_u not adjacent to v , implying that v must be assigned a different value of this property. $\square$

Finally, the following lemma elaborates on the relationship between unconstrained edges in a k -UNO graph and their UNO assignments:

Lemma 8. Let $e = uv$ be an unconstrained edge of a k -UNO graph G . Then, in any UNO assignment of G , u and v either share both properties or they share one and differ in the other, such that the different values are only shared along other unconstrained edges.

Proof. Let M be the unique maximal clique containing u and v . If a vertex $w \in M$ is adjacent to a vertex x outside M then the property P shared by w and x forces the other property P' to be uniform across M . However, the value of P is not yet fixed for the unconstrained vertices of M .

Suppose this value appears elsewhere in the graph—say, u and y have the same value of P . Then, u and y are adjacent. Since M is the only maximal clique containing unconstrained u , y must also belong to M . Hence, y shares P' with u and w .

Now, suppose y has a neighbor z not in M . Then, z would share P but not P' with y , which implies that z is adjacent to u —a contradiction either to u being unconstrained or to z not being in M . Hence, any neighbor of y is in M , and uy is an unconstrained edge. $\square$

The structural differences between unconstrained, semi-constrained, and forced edges identified in Lemmas 7 and 8 are illustrated in Figure 2. The example is based on a K_6 with two additional vertices x and y forming triangles $\{v_1, v_2, x\}$ and $\{v_5, v_6, y\}$. The large clique K_6 determines property a , while the two smaller maximal cliques determine property b . In this configuration, edges v_1v_2 and v_5v_6 are forced, since each lies in two maximal cliques and the endpoints have identical open neighborhoods. Edge v_3v_4 is unconstrained, since both endpoints lie in a unique maximal clique (K_6) and their common neighbors form a clique. All the remaining edges are semi-constrained.

Lemma 8 allows us to characterize the set of possible k -values for which a given graph is a k -UNO graph:

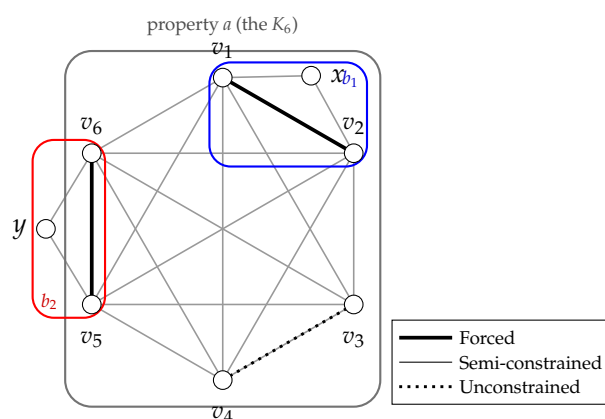


Figure 2. Example graph (K_6 with two extra triangles) illustrating forced, semi-constrained, and unconstrained edges.

Theorem 4. Let G be a k -UNO graph and $f : G \rightarrow G(C)$ its UNO assignment of cards in C , and let $\kappa(C)$ denote the maximum multiplicity of a card in C . Then, the following hold:

- (i) If any two endpoints of any unconstrained edge differ in a property then f is an assignment with the smallest $\kappa_{\min} := \kappa(C)$;
- (ii) If any two endpoints of any unconstrained edge are assigned equal properties then f is an assignment with the largest $\kappa_{\max} := \kappa(C)$;
- (iii) If $\kappa_{\min} \leq k \leq \kappa_{\max}$ then there exists a multiset of cards C and an UNO assignment $f_k : G \rightarrow G(C)$, such that $\kappa(C) = k$.

Proof. For Theorem 4 (i), suppose any two endpoints of an unconstrained edge are assigned different properties, and let c be a card in C with maximum multiplicity $\kappa_{\min}$. Then, there are $\kappa_{\min}$ vertices assigned card c . By assumption, they lie in the intersection of the two maximal cliques corresponding to the property values. Lemma 1 concludes the claim.

For Theorem 4 (ii), suppose for a contradiction that any two endpoints of an unconstrained edge share both properties, and let f', C' be UNO assignments of G with $\kappa(C') > \kappa_{\max}$. Let $c \in C'$ be a card of maximum multiplicity $\kappa(C') > \kappa_{\max} \geq 1$. Let u, v be two vertices assigned card c , and let M be a maximal clique of G containing edge uv . If uv is unconstrained, M is the only clique containing uv . By Lemma 8, any other vertex assigned card c is also in M , and the edges connecting them are unconstrained—a contradiction.

Therefore, uv must be constrained, and there is another maximal clique M' containing uv . By Lemma 1, the vertices in $M \cap M'$ are assigned the same card in any UNO assignment, contradicting $\kappa(C') > \kappa_{\max}$.

For Theorem 4 (iii), let f, C be an UNO assignment of G with $\kappa(C) = \kappa_{\max}$. If there exists a card c of maximum multiplicity $\kappa_{\max}$ such that an edge assigned c is forced then that edge by Lemmas 1 and 4 lies in two maximal cliques, as do all other endvertices that are assigned the same card. Then, $\kappa_{\min} = k = \kappa_{\max}$, and we have finished. Otherwise, proceed by induction for each card c of multiplicity $k + 1$, pick an unconstrained edge and a property shared by its endpoints and their clique M , and assign a new value to the other property not used elsewhere in the graph. This creates a new card with no adjacency outside M .

After applying this to each such card, we reduce the maximum multiplicity to k and obtain the desired UNO assignment. $\square$

4. Algorithms

For the algorithms, we define an extension of the UNO assignment as follows:

Definition 3. Let $G(V, E)$ be a graph. A partial UNO assignment of G is a tuple,

$$(G, A, B, C, p, q, s, Q),$$

where the following applies:

- (p.i) A is a set of numbers, B a set of colors, and $C \subseteq A \times B$ is a set of cards. Let $A_\epsilon := A \cup \{\epsilon\}$ and $B_\epsilon := B \cup \{\epsilon\}$ be extended sets of values, with ϵ indicating that a property is unassigned.
- (p.ii) We can say that $p : V(G) \cup E(G) \rightarrow A_\epsilon \times B_\epsilon$ assigns properties from A and B to vertices and edges of G .
- (p.iii) We can say that $q : A \cup B \cup C \rightarrow \mathbb{N}$ counts the occurrences of numbers from A , colors from B , and cards from C in the assignment p restricted to vertices of G .
- (p.iv) We can say that $s : V(G) \rightarrow \mathbb{Z}_3$ indicates the state of a vertex in the partial assignment p , which is either untouched (0), enqueued (1), or finalized (2).
- (p.v) Q is the queue used to process the vertices.

A partial UNO assignment is valid if the following holds:

- (v.i) Whenever two vertices share an assigned property, they are adjacent;
- (v.ii) Whenever a property is assigned to an edge, it is also assigned to both its endpoints;
- (v.iii) Any vertex that is p -assigned a property is either enqueued or finalized;
- (v.iv) Any finalized vertex is p -assigned both a number $a \in A$ and a color $b \in B$, and the card (a, b) is in C ;
- (v.v) For any finalized vertex v , any edge incident to v and any vertex adjacent to v is assigned a property;
- (v.vi) For any property $x \in A \cup B \cup C$, there are $q(x)$ vertices p -assigned the property x ;
- (v.vii) Any finalized vertex $v \in V(G)$ assigned a card (a, b) is for each property $x \in \{a, b, (a, b)\}$ adjacent to $q(x)$ vertices assigned the property x ;
- (v.viii) A vertex v is enqueued ($s(v) = 1$) if and only if $v \in Q$.

A partial UNO assignment is completed if the following holds:

- (c.i) Each vertex is assigned both a color and a number;
- (c.ii) Whenever two adjacent vertices share a property, the edge between them shares that property as well;
- (c.iii) Any vertex is finalized ($s(v) = 2$).

A valid partial UNO assignment $\alpha' = (G, A', B', C', p', q', s', Q')$ of a graph G is an extension of another valid partial UNO assignment $\alpha = (G, A, B, C, p, q, s, Q)$ of the same graph G , if the following holds:

- (e.i) $A \subseteq A', B \subseteq B', C \subseteq C'$;
- (e.ii) $\forall x \in V(G) \cup E(G)$, whenever $p(x) \neq p'(x)$, then $p'(x)$ matches $p(x)$ in any assigned property of $p(x)$;
- (e.iii) $\forall y \in A \cup B \cup C$, $q(y) = q'(y)$, i.e., q' does not change vertex-assignments of properties already assigned by α ;
- (e.iv) $\forall v \in V(G)$, $s(v) \leq s'(v)$, i.e., the status of any vertex does not decrease.

Our algorithm starts with an empty partial assignment (valid by vacuous conditions) and maintains validity until a completed partial assignment is obtained. The following theorem summarizes the key ideas, as well as listing all the tools needed to prove the correctness. To aid readability, we precede the theorem with a commented pseudocode sketch that captures the main intuition of the algorithm, while omitting the technical details, which are elaborated in the proof.

Pseudocode overview of the algorithm:

1. If no induced path of length 2 exists, the graph is a clique.
 - 1.1 Proceed as sketched in the proof of Theorem 5 (a.iii) for cliques.
2. Suppose an induced path xvy of length two is found.
 - 2.1 Assign a number a_1 to v , x , and the neighbors of v adjacent to x . This fixes the property defining the clique shared by v and x . Enqueue all these neighbors of v .
 - 2.2 Assign a color b_1 to v , y , and all their common neighbors. This fixes the property of the other clique containing v . Enqueue all these neighbors of v as well.
3. While there is an enqueued vertex v , pop it from the queue.
 - 3.1 Assign to it the other property, as well as to all its neighbors that have this property not yet assigned, and enqueue them as discussed in Theorem 5 (a.iii). This yields the next partial UNO assignment in the sequence claimed to exist by Theorem 5.
 - 3.2 If the assignment process encounters a violation of the desired structural properties, then (as established in the proof) the input graph was not a k -UNO graph.

This pseudocode illustrates how the algorithm extends a partial UNO assignment one clique at a time, maintaining validity throughout. The technical details of correctness and time complexity are given in the following theorem.

Theorem 5. Let G be a connected simple graph on n vertices and $\alpha = (G, A, B, C, p, q, s, Q)$ its partial UNO assignment:

- (a.i) For any graph G , the assignment that assigns ε to each property of any vertex and edge is a valid UNO assignment. We call it the empty UNO assignment.
- (a.ii) A valid completed assignment α of G exists if and only if G is a k -UNO graph for some integer k .
- (a.iii) There exists a sequence of $n + 1$ valid partial UNO assignments starting with the empty UNO assignment of G and concluding with a valid completed UNO assignment of G , such that each element of the sequence is an extension of the previous element if and only if G is a k -UNO graph for some integer k .
- (a.iv) Given a function $\alpha : E(G) \rightarrow \{\text{true}, \text{false}\}$ indicating whether the endvertices of G share the common set of neighbors, the sequence from item (a.iii) can be constructed or its existence disproved in time $O(|E(G)|)$.
- (a.v) The function α from item (a.iv) can be found in time $O(\Delta(G)|E(G)|)$, where $\Delta(G)$ is the maximum degree of G .
- (a.vi) By setting $\alpha : E(G) \rightarrow \{\text{false}\}$, the algorithm from (a.iv) constructs the sequence from item (a.iii) if and only if G is a 1-UNO graph. Its running time is still $O(|E(G)|)$.

Proof. For (a.i), define $A = B = C = \emptyset$, p is a constant assigning ε to any property of any vertex and edge, q is the null function, s is the constant function assigning 0 to any vertex, and Q is the empty queue. Properties (v.i) to (v.viii) are easily verified, as they are mostly void; hence, the empty partial UNO assignment is valid.

For (a.ii), let us first assume that G is a k -UNO graph as witnessed by an UNO assignment f of cards $C \subseteq A \times B$. For A, B, C , (p.i) applies. Let p be as f on $V(G)$, and let $p(e)$ assign to the edge the property shared by its endvertices, satisfying (p.ii) and (v.ii). Also (v.i) is satisfied, as f is an UNO assignment. Note that this also implies (c.i) and (c.ii). Let $s : V(G) \rightarrow \mathbb{Z}_3$ assign each vertex of G the constant 2, declaring it finalized. Through p inheriting properties of the UNO assignment f , (v.iv) is satisfied, as are (v.v) and (c.iii). There is a unique and obvious way of defining $q : A \cup B \cup C \rightarrow \mathbb{N}$ to satisfy (v.vi). Moreover, p being induced by an UNO assignment f , together with properties (v.i) and (v.ii), implies (v.vii). To satisfy (p.v), (v.iii), and (v.viii), let Q be the empty queue. The above establishes all the required properties for the claim that the thus-defined partial UNO assignment $\alpha = (G, A, B, C, p, q, s, Q)$ is a completed valid partial UNO assignment of G .

For the converse direction, suppose that a completed valid partial UNO assignment $\alpha = (G, A, B, C, p, q, s, Q)$ is given. It is easy to use properties (p.i), (p.ii), (v.i), (v.ii), (v.iii), (c.iii), (v.iv) to verify that p restricted to $V(G)$ is an UNO assignment, concluding the proof of (a.ii).

We continue with (a.iii), which represents the core of this theorem. Observing that the last element of the sequence is a completed valid partial UNO assignment of G , the existence of the sequence implies that G is a k -UNO graph by (a.ii).

For the rest of the proof of (a.iii), we assume that G is a k -UNO graph and construct the sequence. We first handle cliques, which can be assigned a single card. While these may be considered trivial, their clarity is advantageous for the reader to become familiar with the concepts, but for fast reading it may be skipped.

Let G be a clique on n vertices. Then, any edge of G is unconstrained. Hence, we pick a number a and a color b and assign them to the first vertex v_1 , its incident edges, and all adjacent vertices (i.e., all vertices). This ascertains (p.i), (p.ii), (v.i), (v.ii), and (c.i).

We set v_1 to be finalized and all other vertices to be enqueued, assuring (p.iv), (v.iii), and (v.v). We set q to be the constant n on number a , color b , and card (a, b) , assuring (p.iii), (p.v), (v.vi), and (v.vii). As only the edges incident with v_1 are assigned properties, v_1 is the only finalized vertex, and all other vertices are enqueued, assuring (v.viii). The resulting partial UNO assignment is the first non-empty assignment in the sequence, and it extends the previous empty partial UNO assignment.

We proceed iteratively by popping the i -th vertex from queue Q , p -assigning its non-assigned incident edges properties a and b , thus maintaining (v.ii) and changing its status from enqueued to finalized, maintaining (v.vi) and (v.vii). Each such change maintains all the properties not mentioned and produces a new valid partial UNO assignment of G that extends the previous one by construction. As there are n vertices, this process yields a sequence of $n + 1$ partial UNO assignments of G . Any element of the sequence has one more finalized vertex than the previous, and the last one has all its vertices finalized. We claim that the last element of the sequence is not just valid but also completed. Validity follows by induction, as does the property (c.i), which has been established already at the first vertex; (c.iii) follows, as each of the vertices has been popped from the queue Q ; (c.ii) follows from (c.iii), as each edge has been processed as incident to its first finalized endvertex.

For proving the rest of (a.iii), we assume that G is not a clique. We use properties from Section 3 to construct the required sequence of extending valid partial UNO assignments. As G is connected and not a clique it has an induced path xvy , such that x and y are not adjacent. By Lemma 3, both edges xv and vy are constrained. As x and y are not adjacent, they are in different maximal cliques M_x and M_y of G , the first containing x and the other y . If an UNO assignment of G exists then vertices of one share the number and vertices of the other share the color, while vertices of both share the card with v . Moreover, these are all the neighbors of v in a k -UNO graph G .

Thus, we can produce the first non-empty partial UNO assignment, as follows: choose a number a and a color b . Iterate over neighbors of v , enqueue each of them, and set $s(u) = 1$. For a common neighbor u of v and x , set $p(u) = (a, \cdot)$, $p(uv) = (a, \cdot)$, where $\cdot$ denotes the previous assignment of the property (in this case, ε , but later this will be different). For a common neighbor u of v and y , set $p(u) = (\cdot, b)$, $p(uv) = (\cdot, b)$. Note that in this case, $\cdot$ can represent either ε or a , the latter for u adjacent to all of x , y , and v . Should there be any other neighbor of v , the three maximal cliques of G containing v witness that G is not a k -UNO graph by Theorem 3. Set v to be finalized by $s(v) = 2$. No other vertex will be assigned a or b , so we can set $q(a)$, $q(b)$, and $q(a, b)$. By this procedure, we have established properties (p.i)–(p.v) and (v.i)–(v.viii). As we were modifying the initial empty partial UNO assignment, properties (e.i)–(e.iv) are clear; thus, we have obtained its extension.

We proceed in a similar fashion, by popping a vertex v from Q one after another. As enqueued, by (v.iii), v has at least one property r assigned value t . Let r' be the other property and suppose it has been assigned value t' . Now, for any neighbor u of v :

- (a) assign any property shared by $p(u)$ and $p(v)$ to $p(uv)$;
- (b) ascertain if $p(u)$ has at least one of the two properties equal as $p(v)$;
- (c) count the neighbors of v assigned t , t' , and both.

If r' has not yet been assigned in $p(v)$, pick a value $t' \notin A \cup B$ for the property r' , and, for any neighbor u of v :

- (i) augment p , A , B , C to assign t' as the value of r' for v ;

- (ii) count whether u has been assigned t , t' , or both;
- (iii) if u is not assigned t as the value of r , augment p and C and q to assign t' to u as well as to the edge uv ;
- (iv) also assign t' if u is assigned t but shares all the neighbors with v ;
- (v) enqueue u if it has just been assigned its first property;
- (vi) verify that each neighbor that has r' assigned different from t' has the property r equal to t ;
- (vii) if t' has been assigned, verify that the count of neighbors assigned t , t' , and both equals the respective values of q ;
- (viii) if t' has been newly chosen, verify that the count of neighbors assigned t is equal to $q(t)$ and assign $q(t')$ and $q(c)$ for c the card with properties equal to t, t' ;
- (ix) finalize the vertex v .

There are four verifications that may fail. Through (b) or (vi), there may be a neighbor u of v assigned property r' the value $t' \neq t$ and assigned r a property $t \neq t'$. This neighbor is in three distinct maximal cliques: two have assigned their two values and the third one contains v . By Lemma 2, G is not a k -UNO graph, implying the non-existence of the sequence we are aiming for by (a.ii) and the already proven direction of (a.iii). Failing through (vii) or (viii), v is not adjacent to a vertex previously assigned t, t' , or both, implying that the vertex assigning this property to v is in three different maximal cliques, leading to the same conclusion.

Assuming the process did not fail, we observe that (p.i)–(p.v) are maintained; hence, the new assignment is a partial UNO assignment. Property (v.i) is maintained by (iii), (iv), and (vi). Property (v.ii) is maintained by (a), (iii). Property (v.iii) follows from (v) and (ix). Property (v.iv) follows from (iii), (iv), and (ix). Property (v.v) follows from (a), (b), (iii), and (iv). Property (v.vi) follows from (viii). Property (v.vii) follows from (b), (ii), (vii), and (viii). Property (v.viii) follows from popping v out of Q , as well as (v). Together, these imply that the newly obtained assignment is valid.

Next, we show that the augmented assignment is an extension of the previous assignment. Property (e.i) is established in (iii). Property (e.ii) follows from (a), (iii), and (iv). Property (e.iii) follows from (viii). Property (e.iv) follows from (v) and (ix).

We proceed to show that if Q is empty after finalizing v then the last valid partial UNO assignment is also completed. Suppose that there is a vertex u of G not finalized. Since Q is empty, u has not been enqueued. By (v.iii), u is assigned no property. By (v.v), u is not adjacent to any finalized vertex. As any neighbor of any finalized vertex has been enqueued, G is not connected, a contradiction implying (c.iii), that all vertices of G are finalized. Then, (v.iv) implies (c.i) and (v.v) implies (c.ii). This concludes the proof of (a.iii).

To establish (a.iv), observe that the described algorithms require constant-time operations for each neighbor of each vertex. Therefore, their complexity is $O(|E(G)|)$. Also, the induced path of length 2 can be found by examining the edges of the graph, resulting in the complete time $O(|E(G)|)$.

Claim (a.v) is established by the following: first, convert the adjacency lists of G into hashed sets in time $O(|E(G)|)$. Then, for each edge uv , let N_u and N_v be the corresponding neighborhoods. Sets $N'_u := N_u \setminus \{v\}$ and $N'_v := N_v \setminus \{u\}$ can be computed in place in amortized constant time, as this is the complexity of element deletion (and addition) for hashed sets. Then, the equality of N'_u and N'_v can be verified in $O(\Delta(G))$, and sets N_u, N_v restored in constant time. Altogether, the running time is $O(\Delta(G)|E(G)|)$, as claimed.

Finally, (a.vi) follows from Lemma 6. Fixing $\alpha(e)$ to false implies that the algorithm will not assign equal properties to any two endvertices of an edge sharing the neighbors. This will result in endvertices of unconstrained edges receiving one same and one different property. For vertices of fixed edges, they will be assigned only one property when

enqueued, and an attempt at assigning the second property when popped from the queue will violate condition (b) or will violate (vi) for some neighbor when assigned the second property. Thus, with modified α the algorithm will fail as earlier and, in addition, for k -UNO graphs with $k \geq 2$. The running time remains the same. $\square$

Note that by step (iv), Lemma 3, and Theorem 4, the UNO assignment obtained by (a.iii) yields the UNO assignment with the maximal multiplicity of a card. Using the approach of Theorem 4 (iii), we are able to prove the following:

Corollary 1. *Let G be a graph. Given the function $\alpha : E(G) \rightarrow \{\text{true}, \text{false}\}$ indicating whether the endvertices of e share the common set of neighbors, we can either prove that G is not a k -UNO graph for any integer k or provide the full interval of values k , for which G can be realized as a line graph of a bipartite multigraph with multiplicity k , in time $O(|E(G)|)$.*

Proof. We construct a completed valid partial UNO assignment, $\mathfrak{c} = (G, A, B, C, p, q, s, Q)$ of G , using the algorithm from Theorem 5, or we disprove its existence in the required time. By the observation motivating the theorem, $\mathfrak{c}$ satisfies the properties of Theorem 4 (ii). To conclude the proof, we establish that $\mathfrak{c}$ can be augmented to a completed valid partial UNO assignment $\mathfrak{c}' = (G, A', B', C', p', q', s', Q)$ satisfying Theorem 4 (i) in time $O(|E(G)|)$.

For each vertex v , we examine the following: If there is an edge incident with v that is p -assigned just one of the properties, that property of v is fixed, as changing it would result in an edge whose endvertices would share no property. In $O(d_v)$ steps, we therefore determine whether v is unconstrained, i.e., whether it has a property that can be altered, as the other property is shared by all its neighbors. Examining $q(p(v))$ shows whether the card of v is assigned to more than one vertex. If both conditions are met, we change the property of v and augment the values on its adjacent edges, as well as updating q . For the full graph, this can be completed in $O(|E(G)|)$ time, as required. Once completed, the maximal value of q restricted to C is the minimum value of k , for which G is a k -UNO graph. $\square$

Corollary 2. *The following problems can be solved in linear time for graphs of bounded maximum degree $\Delta(G)$: $O(\Delta(G)|E(G)|)$ on line graphs of bipartite multigraphs: (a) maximum clique, (b) minimal clique cover, (c) maximum weighted clique, (d) chromatic number, (e) independence number.*

Proof. Let G be a line graph of a bipartite multigraph. A completed valid partial UNO assignment $\mathfrak{c} = (G, A, B, C, p, q, s, Q)$ of G can be obtained in time $O(\Delta(G)|E(G)|)$.

For (a), observe that $\max\{q(x) \mid x \in A \cup B\}$ gives the size of a maximal clique of G .

For (b), observe that each number and each color induces a clique in G , and that two cliques assigning different values of the same property are disjoint. Hence, $\min(|A|, |B|)$ gives the minimum clique cover.

For (c), summing the weights for each of the cliques given by the properties is performed in $O(|E(G)|)$.

For (d), Maffray and Reed proved that line graphs of bipartite multigraphs contain no odd holes [22] and are perfect by the strong perfect graph theorem [23], implying that (a) also gives the chromatic number by the strong perfect graph theorem.

Claim (e) follows by duality from (b), as the size of a maximum independent set equals the minimum number of cliques covering all vertices in perfect graphs, due to the weak perfect graph theorem by Lovasz [24]. In order to actually construct a maximum independent set of G , we observe that it corresponds to a matching in the bipartite multigraph H of which G is the line graph. Observing that the numbers and colors are the vertices of this graph and their assignment to vertices of G describes the edges, H has $r = |A| + |B|$ vertices and $|V(G)|$ edges. Given that maximum matching in bipartite multigraphs can

be solved in $O(n^2\sqrt{n})$ by reduction to a max flow problem [25], this yields a $O(r^2\sqrt{r})$ algorithm for constructing the maximum independent set of G , whose size has previously been found in time $O(\Delta(G)|E(G)|)$. $\square$

In a further reflection on Theorem 5, we observe that claim (a.vi) implies that our algorithm recognizes 1-UNO graphs in linear time $O(|E(G)|)$, as the constant $\alpha \equiv \text{false}$ requires no additional computation. The problems of Corollary 2 are, hence, solved in $O(|E(G)|)$, i.e., linear time for line graphs of bipartite graphs, which is currently only known to be polynomial [26].

The previous paragraph identifies the most efficient subcase of our approach that yields linear algorithms for 1-UNO graphs, which do not need the pre-computing of the same-neighborhood-endvertices function α on the edges of G . We hereby emphasize that the least efficient subcase arises when this function is needed for dense graphs with $\Delta(G)$ being proportional to $|V(G)|$. In this case, the exact algorithm complexity becomes $O(|V(G)| \cdot |E(G)|)$. As the number of edges is proportional to the squared number of vertices, the complexity in this case equals the complexity of the previously best-known algorithm of Peterson, which is $O(|V(G)|^3)$.

5. Conclusions and Open Problems

We conclude with an overview of the results augmented with a list of open problems related to those results.

While bipartite simple graphs can be uniquely reconstructed from their line graphs, the same is not true for bipartite multigraphs, as illustrated by cliques. For G , a line graph of a bipartite multigraph H , we propose a characterization of the edges of G into forced, semi-constrained, and unconstrained edges. Forced edges of G connect vertices representing parallel edges of H , semi-constrained edges connect vertices representing adjacent but non-parallel edges of H , and for unconstrained edges G does not provide sufficient information to decide. We then define a valid partial assignment of H edges to G vertices, we show that an empty partial assignment is connected to a completed assignment in the space of valid partial assignments if and only if G is a line graph of a bipartite multigraph, and we exhibit an algorithm of time complexity $O(\Delta(G) |E(G)|)$ that finds the respective path by iteratively increasing a partial assignment one vertex per iteration.

The accompanying data structure of the algorithm also solves the maximum clique, minimum clique cover, maximum weighted clique, chromatic number, and independence number problems on line graphs of bipartite multigraphs. We prove that given the assumption that the original bipartite graph is simple, these problems are solved in linear time, an improvement over current best polynomial algorithms as listed by `graphclasses.org` [26]. Moreover, solving one of the following problems in linear time would extend the linearity from line graphs of simple bipartite graphs to line graphs of bipartite multigraphs:

Problem 1.

- (i) Given a graph G , can the function $\alpha : E(G) \rightarrow \{\text{true}, \text{false}\}$ that identifies vertices with the same set of neighbors be constructed in time $O(E(G))$?
- (ii) Given a graph G , can we decide in $O(d(v))$ time whether a vertex failing through violating (b) or (vi) in the proof of Theorem 5 (a.iii) is failing due to a forced edge or due to a different structural violation of the graph G ?
- (iii) Given a graph G , can we recognize for all edges whether each edge is either fixed or unconstrained in time $O(|E(G)|)$?
- (iv) Can we recognize either all-fixed or all-unconstrained edges in time $O(|E(G)|)$?

Among these problems, we believe that Problem 1 (i) is the most promising direction for future work, as it directly addresses the main source of non-linearity in our recognition algorithm. A positive resolution would extend our linear-time results from bipartite simple graphs to bipartite multigraphs. Next, Problem 1 (ii) also holds considerable potential: refining the handling of failing vertices may yield new algorithmic insights, even if Problem 1 (i) remains unresolved. Problems 1 (iii) and 1 (iv) appear somewhat less ambitious but remain valuable: solving them would still enable linear-time algorithms in restricted settings. Altogether, we see Problem 1 (i) as the most impactful path forward, with the other problems providing intermediate steps and alternative approaches.

Our work also improves the $O(|V(G)|^3)$ algorithm for recognizing line graphs of bipartite multigraphs by Peterson [2], who focused on describing classes of line graphs of two-partite graphs and multigraphs in terms of forbidden induced subgraphs, and who called such graphs 2D gridline graphs. In addition, our results build upon the work of Demaine et al., who called such graphs UNO graphs, and the earlier independent work of the following: Hedetniemi, who referred to them as graphs of (0,1)-matrices; Cook et al., who named them adjacency graphs; and Gurvich and Temkin, where the English translation referred to them as checked graphs.

Extending the known results, it may be interesting to see how the insights of this and related papers would extend to either (blow-ups of) p D gridline graphs or to p D k -UNO graphs. It seems that the formalized partial UNO assignment may be of relevance in this research.

Thus, illustrated wide interest in line graphs of bipartite multigraphs reflects their applicability in several areas of combinatorics, optimization, and applied graph theory, including scheduling, list coloring, stable marriage modeling, timetabling, robotics path planning, and the enumeration of independent sets via rook polynomials, where the algorithms and the newly introduced partial assignment space may be of interest. Note that in some practical applications, such as scheduling, there are usually a bounded number of machines and tasks available, so the practical graphs may be considered bounded degree. In robotics path planning, on the other hand, the number of points from which we are able to reach a turning point may be large, but perhaps bounded degree graphs can be achieved through limiting their number in one direction and picking a robot-nearest model point when seeking the solution.

Author Contributions: Conceptualization, D.B. and J.J.; methodology, D.B. and J.J.; validation, D.B. and J.J.; formal analysis, D.B. and J.J.; investigation, D.B. and J.J.; resources, D.B. and J.J.; writing—original draft preparation, D.B. and J.J.; writing—review and editing, D.B. and J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This work was in part supported by the Slovenian Research and Innovation Agency (ARIS) under the grants P1-0297 and P5-0433.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The author Drago Bokal declares no any direct or potential conflicts of interest with DataBitLab d.o.o.

References

1. Demaine, E. D.; Demaine, M. L.; Harvey, N. J. A.; Uehara, R.; Uno, T.; Uno, Y. UNO is hard, even for a single player. *Theor. Comput. Sci.* **2014**, *521*, 51–61. [[CrossRef](#)]
2. Peterson, D. Gridline graphs: A review in two dimensions and an extension to higher dimensions. *Discrete Appl. Math.* **2003**, *126*, 223–239. [[CrossRef](#)]

3. Dey, P.; Goyal, P.; Misra, N. UNO gets easier for a single player. In Proceedings of the 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, 1–3 July 2014; pp. 147–157.
4. Lampis, M.; Makino, K.; Mitsou, V.; Uno, Y. Parameterized edge hamiltonicity. *Discrete Appl. Math.* **2018**, *248*, 68–78. [\[CrossRef\]](#)
5. Hedetniemi, S. T. Graphs of $(0, 1)$ -matrices. In *Recent Trends in Graph Theory*; Lecture Notes in Mathematics; Capobianco, M.; Frechen, J. B.; Krolík, M., Eds.; Springer: Berlin/Heidelberg, Germany, 1971; Volume 186, pp. 157–171.
6. Cook, C. R.; Acharya, B. D.; Mishra, V. Adjacency graphs. *Congr. Numer.* **1974**, *10*, 317–331.
7. Gurvich, V. A.; Temkin, M. A. Checked perfect graphs. *Sov. Math. Dokl.* **1992**, *326*, 227–232; English translation in *Dokl. Math.* **1993**, *46*, 248–253.
8. Hemminger, R. L.; Beineke, L. W. Line graphs and line digraphs. In *Selected Topics in Graph Theory*; Beineke, L. W., Wilson, R. J., Eds.; Academic Press: Cambridge, MA, USA, 1978; pp. 271–305.
9. Staton, W.; Wingard, G. C. On line graphs of bipartite graphs. *Util. Math.* **1998**, *53*, 183–187.
10. Roussopoulos, N. D. A $\max\{m, n\}$ algorithm for determining the graph H from its line graph G . *Inf. Process. Lett.* **1973**, *2*, 108–112. [\[CrossRef\]](#)
11. König, D. Über Graphen und ihre Anwendungen auf Determinantentheorie und Mengenlehre. *Math. Ann.* **1916**, *77*, 453–465. (In German) [\[CrossRef\]](#)
12. Maffray, F. Kernels in perfect line-graphs. *J. Combin. Theory Ser. B* **1992**, *55*, 1–8. [\[CrossRef\]](#)
13. Tucker, A. Coloring perfect $(K_4 \setminus e)$ -free graphs. *J. Combin. Theory Ser. B* **1987**, *42*, 313–318. [\[CrossRef\]](#)
14. Pecher, A.; Wagler, A. K. Clique and chromatic number of circular-perfect graphs. *Electron. Notes Discrete Math.* **2010**, *36*, 199–206. [\[CrossRef\]](#)
15. Grötschel, M.; Lovász, L.; Schrijver, A. Polynomial algorithms for perfect graphs. *Ann. Discrete Math.* **1984**, *21*, 325–356.
16. Kloks, T. $K_{1,3}$ -free and W_4 -free graphs. *Inf. Process. Lett.* **1996**, *60*, 221–223. [\[CrossRef\]](#)
17. Galvin, F. The list chromatic index of a bipartite multigraph. *J. Combin. Theory Ser. B* **1995**, *63*, 153–158. [\[CrossRef\]](#)
18. Roberts, F. S. *Applied Combinatorics*; Prentice-Hall: Englewood Cliffs, NJ, USA, 1984.
19. Dinu, R.; Navarra, F. On the rook polynomial of grid polyominoes. *arXiv* **2023**, arXiv:2309.01818. [\[CrossRef\]](#)
20. AbuJabal, N.; Rabie, T.; Baziyad, M.; Kamel, I.; Almazrouei, K. Path planning techniques for real-time multi-robot systems: A systematic review. *Electronics* **2024**, *13*, 2239. [\[CrossRef\]](#)
21. Diestel, R. *Graph Theory*, 4th ed.; Graduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 2010; Volume 173.
22. Maffray, F.; Reed, B. A. A description of claw-free perfect graphs. *J. Combin. Theory Ser. B* **1999**, *75*, 134–156. [\[CrossRef\]](#)
23. Chudnovsky, M.; Robertson, N.; Seymour, P.; Thomas, R. The strong perfect graph theorem. *Ann. Math.* **2006**, *164*, 51–229. [\[CrossRef\]](#)
24. Lovász, L. Normal hypergraphs and the perfect graph conjecture. *Discrete Math.* **1972**, *2*, 253–267. [\[CrossRef\]](#)
25. Even, S.; Tarjan, R. E. Network flow and testing graph connectivity. *SIAM J. Comput.* **1975**, *4*, 507–518. [\[CrossRef\]](#)
26. de Ridder, H. N. Information System on Graph Classes and Their Inclusions (ISGCI), 2001–2025, Updated 9 July 2025. Available online: https://www.graphclasses.org/classes/gc_251.html (accessed on 10 July 2025).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.