mathematics

MDPI

Article

On the Complexity of Computing a Maximum Acyclic Matching in Undirected Graphs

Samer Nofal

Department of Computer Science, German Jordanian University, Amman 11180, Jordan; samer.nofal@gju.edu.jo

Abstract: The problem of finding a maximum acyclic matching in a simple undirected graph is known to be NP-complete. In this paper, we present new results; we show that a maximum acyclic matching in a given undirected graph (with n vertices and m edges) can be computed recursively with a recursion depth $\mathcal{O}(\ln m)$ in expectation. Consequently, employing a recursive computation of a maximum acyclic matching in a given graph, if the recursion depth meets the expectation $\mathcal{O}(\ln m)$, then a maximum acyclic matching can be computed in time $\mathcal{O}(n^{3.4})$ and space $\mathcal{O}(m \ln m)$. However, for the general case, the complexity of the recursive computation of a maximum acyclic matching is in $\mathcal{O}(n^2 2^m)$ time and in $\mathcal{O}(m^2)$ space.

Keywords: maximum acyclic matching; undirected graph; recursive algorithm

MSC: 05C85

1. Introduction

The matching theory is a major field in combinatorics and graph theory; see, e.g., the work of [1–7]. The focus of this paper is the so-called *maximum acyclic matching* in simple, connected, undirected graphs. Let $G = (V, E)$ be an undirected graph with a set V of n vertices and a set E of m edges. A subset M of E is a *matching* in G if no two edges in M share a common vertex. A matching M in G is *acyclic* if the subgraph of G induced by the set of vertices incident on the edges of M is cycle-free. M is a *maximum* acyclic matching if for all acyclic matchings S in G, $|M| \geq |S|$. It is known that the problem of finding a maximum acyclic matching in an undirected graph is NP-complete, as shown by [8,9].

In this paper, we present new results concerning the time complexity of finding a maximum acyclic matching in an undirected graph. In particular, we show that a maximum acyclic matching in a given undirected graph (with n vertices and m edges) can be computed recursively with a recursion depth $\mathcal{O}(\ln m)$ in expectation. Thus, employing a recursive computation of a maximum acyclic matching in a given graph, if the recursion depth meets the expectation $\mathcal{O}(\ln m)$, then a maximum acyclic matching can be computed in time $\mathcal{O}(n^{3.4})$ and space $\mathcal{O}(m \ln m)$. However, for the general case, the recursive computation of a maximum acyclic matching is in $\mathcal{O}(n^2 2^m)$ time and $\mathcal{O}(m^2)$ space.

The rest of this article is organized as follows: In Section 2, we discuss related work. In Section 3, we discuss a recursive algorithm (Algorithm 1) for computing a maximum acyclic matching in a given undirected graph. In Section 4, we stress the correctness of the recursive algorithm in Theorem 1. In Section 5, we analyze the time and space of the recursive algorithm, where our main results are stated in Theorems 2 and 3. We conclude our article in Section 6.

check for updates

Academic Editor: Andrea Scozzari

Received: 23 January 2025
Revised: 5 March 2025
Accepted: 5 March 2025
Published: 6 March 2025

Citation: Nofal, S. On the Complexity of Computing a Maximum Acyclic Matching in Undirected Graphs. *Mathematics* **2025**, *13*, 889. https://doi.org/10.3390/math13050889

Copyright: © 2025 by the author. 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/).

2. Related Work

Restricted matchings and acyclic matchings in graphs have been extensively studied in the literature. In the following, we overview the existing work on the computation of restricted matchings and acyclic matchings in graphs. Ref. [8] investigated generalized subgraph-restricted matchings in graphs. Ref. [9] examined acyclic matchings in subclasses of bipartite graphs. Ref. [10] studied degenerate matchings and edge colorings. Refs. [11,12] investigated the complexity of minimum maximal acyclic matchings. Refs. [13,14] discussed computing acyclic matching in certain subclasses of graphs. Ref. [15] studied hard and certain tractable cases of the maximum acyclic matching problem. Ref. [16] analyzed minimum maximal acyclic matching in proper interval graphs. Ref. [17] discussed the parameterized complexity of the acyclic matching problem. Ref. [18] presented further parameterized results on acyclic matchings. Ref. [19] offered efficient computation of simplicial homology through acyclic matching. Ref. [20] examined acyclic matching on hypercube networks. Ref. [21] considered a lower bound on the acyclic matching number of subcubic graphs. Ref. [20] researched acyclic matching in ladder graphs. Ref. [22] utilized acyclic matching to enhance the current flow in diamond ladder structures. Ref. [23] studied the complexity of minimum maximal uniquely restricted matching. Ref. [24] considered approximating maximum acyclic matchings by greedy and local search strategies. Ref. [25] thoroughly studied restricted matchings in graphs. Ref. [26] examined acyclic matchings in graphs of bounded maximum degree.

However, the expected depth of the recursion tree of the recursive computation of a maximum acyclic matching has not been studied in the literature. Therefore, our results presented in this paper fill this gap, where we calculate the expected depth of the recursion tree of the recursive computation of a maximum acyclic matching and discuss the consequences for the space complexity and the overall time complexity of the concerned computations.

3. Algorithm

In Algorithm 1, we list our method for finding a maximum acyclic matching in a given undirected graph $G = (V, E)$ with a set V of n vertices and a set E of m edges. We go through Algorithm 1 line by line.

In lines 1–3, we start with an empty maximum acyclic matching M and an empty acyclic matching $\mathcal{X}$; we also create a vertex status mapping $\beta : V \to \{1, 2\}$. For a vertex v, the status $\beta(v) = 2$ is whenever the vertex v is adjacent to a vertex incident on an edge in an acyclic matching $\mathcal{X}$; otherwise, the status of the vertex v is $\beta(v) = 1$. As we start with an empty acyclic matching $\mathcal{X}$, the initial status is $\beta(v) = 1$ for all vertices v. In line 3, we create an edge status mapping $\alpha : E \to \{1, 2, 3\}$, such that, for an edge e, the status $\alpha(e) = 2$ indicates that the edge e is included in an acyclic matching $\mathcal{X}$; the status $\alpha(e) = 3$ implies that the edge e is forbidden to join an acyclic matching $\mathcal{X}$, whereas the status $\alpha(e) = 1$ refers to the initial status of the edge e before being included in or excluded from an acyclic matching $\mathcal{X}$.

In lines 4–7, while there are edges e in the graph with status $\alpha(e) = 1$, randomly select an edge e with status $\alpha(e) = 1$. In line 6, we call routine F (defined at line 8) for constructing an acyclic matching $\mathcal{X}$ passing on an empty acyclic matching $\mathcal{X}$, a vertex status β, an edge status α, and the vertices of the selected edge e. Afterward, in line 7, we change the status $\alpha(e) \leftarrow 3$ to prevent e from joining $\mathcal{X}$ in the subsequent calls for routine F, because the algorithm aims to construct other possible acyclic matchings excluding this edge e.

In line 8, we list the routine F's parameters: an acyclic matching $\mathcal{X}$, a vertex status β, an edge status α, and the edge $\{x, y\}$ that has been chosen to join the acyclic matching $\mathcal{X}$.

In line 9, we add the edge $\{x, y\}$ to the acyclic matching $\mathcal{X}$, and update the edge status $\alpha(\{x, y\}) \leftarrow 2$, the vertex status $\beta(x) \leftarrow 2$ and $\beta(y) \leftarrow 2$.

In lines 10–13, we process every vertex $\varphi \neq y$ adjacent to the vertex x (see lines 10–11), such that we change the status $\beta(\varphi) \leftarrow 2$. In addition, we update the status $\alpha(\{\varphi, x\}) \leftarrow 3$ to prevent this edge from joining the acyclic matching $\mathcal{X}$, adhering to the condition that states matching have no two edges sharing a common vertex. Afterward, in lines 12–13, we check every vertex μ adjacent to the vertex φ. If the edge $\{\mu, \varphi\}$ has a status $\alpha(\{\mu, \varphi\}) = 1$ and the vertex status $\beta(\mu) = 2$, then we prevent the edge $\{\mu, \varphi\}$ from joining the acyclic matching $\mathcal{X}$, to maintain acyclicity by excluding any edge whose vertices are adjacent to other vertices incident on certain edges in $\mathcal{X}$. This is true as we build an acyclic matching where the subgraph of G induced by the vertices incident on the edges of $\mathcal{X}$ is connected; we elaborate further on this issue throughout the rest of the article. In lines 14–17, we repeat operations analogous to the actions of lines 10–13, but for y, the other vertex of e.

Concerning lines 18–21, as long as the set $\{\{x, y\} \mid \alpha(\{x, y\}) = 1\} \neq \emptyset$, we repeatedly perform the following actions. We select (in line 19) an edge $\{x, y\}$ (at random) with status $\alpha\{x, y\} = 1$ and $\beta(x) \neq \beta(y)$. This means we randomly select an edge $\{x, y\}$ for whom it is not decided yet whether to include or exclude it from the acyclic matching $\mathcal{X}$, where only one (not both) of the vertices of the edge $\{x, y\}$ is adjacent to a vertex incident on an edge in the acyclic matching $\mathcal{X}$. This tactic ensures connectivity of the subgraph induced by the vertices incident on the edges of $\mathcal{X}$. In line 20, we call routine F, passing on a copy of the acyclic matching $\mathcal{X}$, a copy of the vertex status β, a copy of the edge status α, and the selected edge $\{x, y\}$. We stress that all invocations for routine F are made by calling it with a *copy* of the passed structures; that is, any changes made in these passed structures within the called instance of routine F do not affect the original copy of the caller instance of routine F. The last action within the while loop is to exclude the edge $\{x, y\}$ from any acyclic matchings that might be constructed in the subsequent rounds.

At last, in line 22, since we search for a maximum acyclic matching, we update $M \leftarrow \mathcal{X}$ if $\mathcal{X}$ is more significant in size than M.

Algorithm 1: Find a maximum acyclic matching, M, of a graph $G = (V, E)$.

```
1  M ← ∅; X ← ∅;
2  let β : V → {1,2} be initially β(i) = 1 for all i;
3  let α : E → {1,2,3} be initially α(i) = 1 for all i;
4  while {{x,y} | α({x,y}) = 1} ≠ ∅ do
5      select at random an edge {x,y} with α({x,y}) = 1;
6      F(X,β,α,x,y);
7      α({x,y}) ← 3;

8  routine F(X,β,α,x,y):
9      X ← X ∪ {{x,y}}; α({x,y}) ← 2; β(x) ← 2; β(y) ← 2;
10     foreach φ ≠ y with {φ,x} ∈ E do
11         α({φ,x}) ← 3; β(φ) ← 2;
12         foreach μ : {μ,φ} ∈ E ∧ α({μ,φ}) = 1 ∧ β(μ) = 2 do
13             α({μ,φ}) ← 3;
14     foreach φ ≠ x with {φ,y} ∈ E do
15         α({φ,y}) ← 3; β(φ) ← 2;
16         foreach μ : {μ,φ} ∈ E ∧ α({μ,φ}) = 1 ∧ β(μ) = 2 do
17             α({μ,φ}) ← 3;
18     while {{x,y} | α({x,y}) = 1} ≠ ∅ do
19         select at random an edge {x,y} with α({x,y}) = 1 and β(x) ≠ β(y);
20         F(X,β,α,x,y);
21         α({x,y}) ← 3;
22     if |X| > |M| then M ← X;
```

Example 1. *Apply Algorithm 1 to a graph G_1 with a vertex set $V = \{a, b, c, d, e\}$ and an edge set $E = \{\{a, b\}, \{b, c\}, \{c, a\}, \{a, d\}, \{b, e\}, \{e, c\}, \{e, d\}\}$. Initially, applying the actions in lines 1–3 in Algorithm 1, the state is*

$$M = \emptyset, \mathcal{X} = \emptyset, \beta = \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)\},$$
$$\alpha = \{(\{a, b\}, 1), (\{b, c\}, 1), (\{c, a\}, 1), (\{a, d\}, 1), (\{b, e\}, 1), (\{e, c\}, 1), (\{e, d\}, 1)\}.$$

Since the set $\{\{x, y\} \mid \alpha(\{x, y\}) = 1\} \neq \emptyset$, in line 5, we randomly select an edge, suppose it is $\{a, b\}$. Then, in line 6, we invoke routine F by calling it with

$$F(\emptyset, \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)\},$$
$$\{(\{a, b\}, 1), (\{b, c\}, 1), (\{c, a\}, 1), (\{a, d\}, 1), (\{b, e\}, 1), (\{e, c\}, 1), (\{e, d\}, 1)\}, a, b)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \emptyset, \mathcal{X} = \{\{a, b\}\},$$
$$\alpha = \{(\{a, b\}, 2), (\{b, c\}, 3), (\{c, a\}, 3), (\{a, d\}, 3), (\{b, e\}, 3), (\{e, c\}, 3), (\{e, d\}, 3)\},$$
$$\beta = \{(a, 2), (b, 2), (c, 2), (d, 2), (e, 2)\}.$$

Applying line 18, since the set $\{\{x, y\} \mid \alpha(\{x, y\}) = 1\}$ is empty, we proceed with applying line 22, and the state of the algorithm's execution becomes

$$M = \{\{a, b\}\}, \mathcal{X} = \{\{a, b\}\},$$
$$\alpha = \{(\{a, b\}, 2), (\{b, c\}, 3), (\{c, a\}, 3), (\{a, d\}, 3), (\{b, e\}, 3), (\{e, c\}, 3), (\{e, d\}, 3)\},$$
$$\beta = \{(a, 2), (b, 2), (c, 2), (d, 2), (e, 2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{a, b\}$ from any subsequent acyclic matching. Now the state of the execution of the algorithm is

$$M = \{\{a, b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a, b\}, 3), (\{b, c\}, 1), (\{c, a\}, 1), (\{a, d\}, 1), (\{b, e\}, 1), (\{e, c\}, 1), (\{e, d\}, 1)\},$$
$$\beta = \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)\}.$$

In a second round of the while loop (at line 4), assume we randomly selected (in line 5) the edge $\{c, a\}$. Thereby, we invoke routine F (in line 6) with the following call:

$$F(\emptyset, \{(a, 1), (b, 1), (c, 1), (d, 1), (e, 1)\},$$
$$\{(\{a, b\}, 3), (\{b, c\}, 1), (\{c, a\}, 1), (\{a, d\}, 1), (\{b, e\}, 1), (\{e, c\}, 1), (\{e, d\}, 1)\}, c, a)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a, b\}\}, \mathcal{X} = \{\{c, a\}\},$$
$$\alpha = \{(\{a, b\}, 3), (\{b, c\}, 3), (\{c, a\}, 2), (\{a, d\}, 3), (\{b, e\}, 3), (\{e, c\}, 3), (\{e, d\}, 3)\},$$
$$\beta = \{(a, 2), (b, 2), (c, 2), (d, 2), (e, 2)\}.$$

Applying line 18, since the set $\{\{x, y\} \mid \alpha(\{x, y\}) = 1\}$ is empty, we proceed with applying line 22 and the state of the algorithm's execution remains

$$M = \{\{a, b\}\}, \mathcal{X} = \{\{c, a\}\},$$
$$\alpha = \{(\{a, b\}, 3), (\{b, c\}, 3), (\{c, a\}, 2), (\{a, d\}, 3), (\{b, e\}, 3), (\{e, c\}, 3), (\{e, d\}, 3)\},$$
$$\beta = \{(a, 2), (b, 2), (c, 2), (d, 2), (e, 2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{c,a\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},1),(\{c,a\},3),(\{a,d\},1),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

In a third round of the while loop (at line 4), assume we randomly selected (in line 5) the edge $\{b,c\}$*. Thereby, we invoke routine F (in line 6) with the following call:*

$$F(\emptyset,\{(a,1),(b,1),(c,1),(d,1),(e,1)\},$$
$$\{(\{a,b\},3),(\{b,c\},1),(\{c,a\},3),(\{a,d\},1),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},b,c)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{b,c\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},2),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

Applying line 18, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\}$ *is empty, we proceed with applying line 22 and the state of the algorithm's execution remains*

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{b,c\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},2),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{b,c\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},1),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

In a fourth round of the while loop (at line 4), assume we randomly selected (in line 5) the edge $\{a,d\}$*. Thereby, we invoke routine F (in line 6) with the following call:*

$$F(\emptyset,\{(a,1),(b,1),(c,1),(d,1),(e,1)\},$$
$$\{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},1),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},a,d)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{a,d\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},2),(\{b,e\},3),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

Applying line 18, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\}$ *is empty, we proceed with applying line 22 and the state of the algorithm's execution remains*

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{a,d\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},2),(\{b,e\},3),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{a,d\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

In a fifth round of the while loop (at line 4), assume we randomly selected (in line 5) the edge $\{d,e\}$*. Thereby, we invoke routine F (in line 6) with the following call:*

$$F(\emptyset,\{(a,1),(b,1),(c,1),(d,1),(e,1)\},$$
$$\{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},1),(\{e,c\},1),(\{e,d\},1)\},d,e)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{d,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},3),(\{e,d\},2)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

Applying line 18, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\}$ *is empty, we proceed with applying line 22 and the state of the algorithm's execution remains*

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{d,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},3),(\{e,d\},2)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{d,e\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},1),(\{e,c\},1),(\{e,d\},3)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

In a sixth round of the while loop (at line 4), assume we randomly selected (in line 5) the edge $\{b,e\}$*. Thereby, we invoke routine F (in line 6) with the following call:*

$$F(\emptyset,\{(a,1),(b,1),(c,1),(d,1),(e,1)\},$$
$$\{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},1),(\{e,c\},1),(\{e,d\},3)\},b,e)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{b,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},2),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

Applying line 18, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\}$ *is empty, we proceed with applying line 22 and the state of the algorithm's execution remains*

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{b,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},2),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{b,e\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},1),(\{e,d\},3)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

In a seventh round of the while loop (at line 4), we select (in line 5) the edge $\{c,e\}$*. Thereby, we invoke routine F (in line 6) with the following call:*

$$F(\emptyset,\{(a,1),(b,1),(c,1),(d,1),(e,1)\},$$
$$\{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},1),(\{e,d\},3)\},c,e)$$

Then, we apply the actions of lines 9–17, and subsequently, the state of the algorithm becomes

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{c,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},2),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

Applying line 18, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\}$ *is empty, we proceed with applying line 22 and the state of the algorithm's execution remains*

$$M = \{\{a,b\}\}, \mathcal{X} = \{\{c,e\}\},$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},2),(\{e,d\},3)\},$$
$$\beta = \{(a,2),(b,2),(c,2),(d,2),(e,2)\}.$$

By completing the execution of this instance of the routine F, we return to line 7; we now exclude the edge $\{c,e\}$ *from any subsequent acyclic matching. The state of the execution of the algorithm is now*

$$M = \{\{a,b\}\}, \mathcal{X} = \emptyset,$$
$$\alpha = \{(\{a,b\},3),(\{b,c\},3),(\{c,a\},3),(\{a,d\},3),(\{b,e\},3),(\{e,c\},3),(\{e,d\},3)\},$$
$$\beta = \{(a,1),(b,1),(c,1),(d,1),(e,1)\}.$$

Referring to line 4, since the set $\{\{x,y\} \mid \alpha(\{x,y\}) = 1\} = \emptyset$*, the execution of the while loop is completed. The algorithm's execution ends with the set* $\{\{a,b\}\}$ *being a maximum acyclic matching in the given graph.*

4. Correctness

Our first lemma stresses that for a given graph $G = (V,E)$, Algorithm 1 employs the vertex status $\beta : V \to \{1,2\}$ to mark if a vertex is adjacent to a vertex incident on an edge in the set $\mathcal{X}$.

Lemma 1. *Throughout the execution of Algorithm 1 on a given graph* $G = (V,E)$*, for a vertex* v *in the graph, the vertex status* $\beta(v) = 2$ *whenever the vertex is adjacent to a vertex incident on an edge included in the set* $\mathcal{X}$*. Otherwise, the vertex status is* $\beta(v) = 1$*.*

Proof. In lines 1–2, the set $\mathcal{X}$ is empty, and the vertex status $\beta(v) = 1$ for all v. Now, we show that whenever we change the status $\beta(v) \leftarrow 2$, this entails that v is adjacent to a vertex incident on an edge in the set $\mathcal{X}$. Referring to line 9 in Algorithm 1, once an edge $\{x,y\}$ joins the set $\mathcal{X}$, we label x and y with $\beta(x) = 2$ and $\beta(y) = 2$, because both vertices are adjacent to a vertex incident on the edge $\{x,y\} \in \mathcal{X}$. In addition, in line 11 (respectively 15), we

label a vertex φ with $\beta(\varphi) = 2$, since the vertex φ is adjacent to the vertex x (respectively y) that is incident on the edge $\{x, y\} \in \mathcal{X}$. □

The following lemma stresses that for a given graph $G = (V, E)$, Algorithm 1 uses an edge status $\alpha : E \to \{1, 2, 3\}$ to indicate if an edge is in or out of a set $\mathcal{X}$.

Lemma 2. *Throughout the execution of Algorithm 1 on a given graph $G = (V, E)$, for an edge e in the graph, the edge status is $\alpha(e) = 2$ if and only if e is in a set $\mathcal{X}$. The edge status is $\alpha(e) = 3$ whenever the algorithm aims to construct a set $\mathcal{X}$ excluding $\{e\}$ or whenever e shares a common vertex with an edge included in a set $\mathcal{X}$, or whenever every vertex of the edge e is adjacent to a vertex incident on an edge in a set $\mathcal{X}$. Otherwise, the edge status is $\alpha(e) = 1$.*

Proof. Initially, the set $\mathcal{X}$ is empty, and the edge status $\alpha(e) = 1$ for all e; see lines 1 and 3 in Algorithm 1. Once an edge $\{x, y\}$ joins the set $\mathcal{X}$, we label it with $\alpha(\{x, y\}) = 2$; see line 9. The algorithm labels an edge $\{x, y\}$ with $\alpha(\{x, y\}) = 3$ in three cases. For the first case, the algorithm aims to construct a set $\mathcal{X}$ excluding $\{x, y\}$ after the algorithm has constructed a set $\mathcal{X}$ including $\{x, y\}$, see lines 19–21. The second case of labeling an edge e_1 with $\alpha(e_1) = 3$ is when there is another edge $e_2 \in \mathcal{X}$ such that e_1 and e_2 share a common vertex, see lines 11 and 15 in Algorithm 1. The third case of labeling an edge $\{x, y\}$ with $\alpha(\{x, y\}) = 3$ requires three conditions:

1. x is incident on an edge included in the set $\mathcal{X}$,
2. y is incident to an edge included in the set $\mathcal{X}$, and
3. the edge status $\alpha(\{x, y\}) = 1$; that is, the edge is not in $\mathcal{X}$;

see lines 13 and 17. □

In the following lemma, we emphasize that Algorithm 1 constructs acyclic matchings for a given graph.

Lemma 3. *Throughout the execution of Algorithm 1 on a given graph $G = (V, E)$, the set $\mathcal{X}$ is acyclic matching in G.*

Proof. Initially, $\mathcal{X}$ is acyclic matching because it is empty; see line 1 in Algorithm 1. Now, we show that whenever we add an edge $\{x, y\}$ to an acyclic matching $\mathcal{X}$, the set $\mathcal{X} \cup \{\{x, y\}\}$ is acyclic matching. Referring to line 19–20, we include an edge $\{x, y\}$ in an acyclic matching $\mathcal{X}$ if the two following conditions are satisfied:

(i) $\alpha(\{x, y\}) = 1$, which means, as shown in Lemma 2, that the vertices x and y are not incident on an edge in the acyclic matching $\mathcal{X}$, and

(ii) $\beta(x) \neq \beta(y)$, which means, as shown in Lemma 2, that only x or only y is adjacent to a vertex incident on an edge in $\mathcal{X}$.

Thus, satisfying (i) ensures that the set $\mathcal{X} \cup \{\{x, y\}\}$ is a matching, whereas satisfying (ii) ensures that the set $\mathcal{X} \cup \{\{x, y\}\}$ is acyclic but also guarantees that the subgraph of G induced by the vertices of the edges in $\mathcal{X}$ is connected. □

We are ready to demonstrate that Algorithm 1 finds a maximum acyclic matching of a given graph.

Theorem 1. *Algorithm 1 finds a maximum acyclic matching M of a given graph $G = (V, E)$.*

Proof. Assume the opposite; that is, M is not acyclic matching or M is acyclic matching but not a maximum one. M being not acyclic matching is impossible, since we showed earlier that in all stages, i, of the execution of Algorithm 1, the set $\mathcal{X}_i$ is acyclic matching, and $M \leftarrow \mathcal{X}$ is the only update on M, see line 22 in Algorithm 1. M being not maximum means

that there is an acyclic matching S overlooked by the algorithm such that $|S| > |\mathcal{X}_i|$ in all stages, i, of the execution of Algorithm 1. We note that at some stage, k, of the execution of Algorithm 1, an acyclic matching $\mathcal{X}_k \subset S$. This requires that there is a set of edges $\{e_1, e_2, \ldots, e_l\} \subset S$, but $\{e_1, e_2, \ldots, e_l\} \cap \mathcal{X}_k = \emptyset$. As every edge $(e_q)_{(1 \leq q \leq l)} \notin \mathcal{X}_k$, there are two cases: (i) the status $\alpha(e_q) = 1$ or (ii) the status $\alpha(e_q) = 3$. For case (i), the status $\alpha(e_q) = 1$ entails that the edge e_q will join the acyclic matching $\mathcal{X}_k$ (see lines 18–19 & 4–5), which contradicts the assumption that the algorithm overlooks the set S. For case (ii), where the status $\alpha(e_q) = 3$, as shown in Lemma 2, this implies one of the following:

1. The edge e_q has already joined the acyclic matching $\mathcal{X}_k$, and now the algorithm intentionally aims to grow $\mathcal{X}_k$ without the edge e_q, see lines 19–21 in Algorithm 1. This contradicts the assumption that S is an acyclic matching overlooked by the algorithm.
2. The edge e_q shares a common vertex with an edge in the acyclic matching $\mathcal{X}_k$ (see lines 11 and 15 in Algorithm 1); thus, $\{e_q\} \cup \mathcal{X}_k$ is not a matching, and subsequently, S is not a matching, since $S \supset \{e_q\} \cup \mathcal{X}_k$, contradicting the assumption that S is an acyclic matching overlooked by the algorithm.
3. The edge $e_q = \{x, y\}$ has its vertex x adjacent to a vertex incident on an edge in the acyclic matching $\mathcal{X}_k$ and the edge e_q has its vertex y adjacent to a vertex incident on an edge in $\mathcal{X}_k$ (see lines 13 and 17). So, $\{e_q\} \cup \mathcal{X}_k$ is cyclic, and subsequently $S \supset \{e_q\} \cup \mathcal{X}_k$ is cyclic, contradicting the assumption that S is a acyclic matching overlooked by the algorithm. For this case, it is important to note that we showed earlier in Lemma 3 that for every stage i of the execution of Algorithm 1, the subgraph of G, induced by the vertices of the edges in the acyclic matching $\mathcal{X}_i$, is connected.

□

5. Complexity

In this section, we calculate the expected recursion depth of Algorithm 1. But let us first discuss the general time and space complexity of Algorithm 1. The nested loop in lines 10–13 (similarly, the loop in lines 14–17) runs in $\mathcal{O}(n^2)$ time. Note that the size of the execution tree of Algorithm 1 is in the order of the number of calls to routine F. For each edge e in the given graph, we call routine F twice, once with $\alpha(e) = 1$ and another time with $\alpha(e) = 3$; see lines 6–7 and 20–21. Thus, the size of the execution tree is $\mathcal{O}(2^m)$. Thus, the overall time of Algorithm 1 is in $\mathcal{O}(n^2 2^m)$.

For the running space of Algorithm 1, we note that at maximum, there are m instances of the routine F that are simultaneously active where every instance holds a copy of the structures: $\mathcal{X}$, α and β. So, the running space of Algorithm 1 is in $\mathcal{O}(m^2)$.

Now, we calculate the expected recursion depth of Algorithm 1 in the following theorem.

Theorem 2. *Let $G = (V, E)$ be a graph with n vertices and m edges. Algorithm 1 runs with a recursion depth $\mathcal{O}(\ln m)$ in expectation.*

Proof. The expected recursion depth of Algorithm 1 is equal to

$$\sum_{j=1}^{m} E(X_j)$$

where $E(X_j)$ is the expected number of times an edge j is selected within a complete path from the root call (for routine F) to the leaf call (for routine F) in the execution tree of Algorithm 1. X_j is a random variable representing the number of times an edge j is selected

within a complete path from the root call (for routine F) to the leaf call (for routine F) in the execution tree of Algorithm 1. Thus, X_j is defined as

$$X_j : \{(\{1,2,\ldots,i\},j) \mid 1 \leq i \leq m\} \to \{0,1\}$$

such that $X_j((\{1,2,\ldots,i\},j)) = 1$ if $j \in \{1,2,\ldots,i\}$ and $X_j((\{1,2,\ldots,i\},j)) = 0$ if $j \notin \{1,2,\ldots,i\}$. Therefore,

$$\begin{aligned} E(X_j) &= \sum_{i=1}^{m} X_j((\{1,2,\ldots,i\},j))P(X_j((\{1,2,\ldots,i\},j)) = 1) \\ &= \sum_{i=1}^{m} \left(\frac{1}{m}\right)\left(\frac{1}{i}\right) = \left(\frac{1}{m}\right)\sum_{i=1}^{m}\left(\frac{1}{i}\right), \end{aligned}$$

where $\frac{1}{m}$ is the probability of having a set of i edges and $\frac{1}{i}$ is the probability of selecting an edge $j \in \{1,2,\ldots,i\}$ from $\{1,2,\ldots,i\}$. Hence, the expected recursion depth is

$$\sum_{j=1}^{m} E(X_j) = \sum_{j=1}^{m}\left(\frac{1}{m}\right)\sum_{i=1}^{m}\left(\frac{1}{i}\right) = \sum_{i=1}^{m}\left(\frac{1}{i}\right).$$

Since

$$\sum_{i=1}^{m}\left(\frac{1}{i}\right) \leq \ln m + 1,$$

it is the case that

$$\sum_{i=1}^{m}\left(\frac{1}{i}\right) \in \mathcal{O}(\ln m) \in \mathcal{O}(\ln n^2) \in \mathcal{O}(\ln n).$$

□

We now state the space and time complexity of Algorithm 1 under the assumption that the recursion depth of Algorithm 1 running on a given graph meets the expectation $\mathcal{O}(\ln m)$.

Theorem 3. *For a given graph $G = (V, E)$ with n vertices and m edges, if the recursion depth of Algorithm 1 meets the expectation $\mathcal{O}(\ln m)$, then the algorithm runs in time $\mathcal{O}(n^{3.4})$ and space $\mathcal{O}(m \ln m)$.*

Proof. As discussed earlier, the running time of Algorithm 1 is the cost of the nested loop 10–13 (and 14–17) multiplied by the size of the execution tree of the algorithm, which is overall $\mathcal{O}(n^2 2^m)$. If the recursion depth of the algorithm is equal to the expected depth that is equal to (see the proof of Theorem 2)

$$\sum_{i=1}^{m}\left(\frac{1}{i}\right) \leq \ln m + 1,$$

then the size of the execution tree becomes $\mathcal{O}(2^{\ln m})$, and hence the running time of the algorithm becomes

$$\mathcal{O}(n^2 2^{\ln m}) \in \mathcal{O}(n^2 2^{2\ln n}) \in \mathcal{O}(n^2 n^{2\ln 2}) \in \mathcal{O}(n^{3.4}).$$

Under the assumption that the recursion depth of Algorithm 1 is equal to the expected depth $\mathcal{O}(\ln m)$, the space complexity of Algorithm 1 becomes $\mathcal{O}(m \ln m)$; recall that the algorithm, in every call to the recursive routine F, makes a $\mathcal{O}(m)$ copy of the passed structures (i.e., $\mathcal{X}$, α, and β). □

6. Conclusions

We presented new results on the complexity of computing a maximum acyclic matching in a given undirected graph. We demonstrated that a maximum acyclic matching in a graph can be computed recursively with a recursion depth $\mathcal{O}(\ln m)$ in expectation. This implies that the complexity of recursively computing a maximum acyclic matching in a graph, such that the recursion depth meets the expectation $\mathcal{O}(\ln m)$, is $\mathcal{O}(n^{3.4})$ time and $\mathcal{O}(m \ln m)$ space. However, for the general case, the time complexity of the discussed recursive computation for a maximum acyclic matching is in $\mathcal{O}(n^2 2^m)$ and the space complexity is in $\mathcal{O}(m^2)$. Further investigations could consider diverse randomized strategies in the computation of a maximum acyclic matching with different probability distributions in particular classes of undirected graphs.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lovász, L.; Plummer, M. *Matching Theory*; AMS Chelsea Publishing Series; AMS Chelsea Pub.: New York, NY, USA, 2009.
2. Gerards, A. Matching. In *Handbooks in Operations Research and Management Science*; Elsevier: Amsterdam, The Netherlands, 1995; Volume 7, pp. 135–224.
3. Yan, J.; Yin, X.C.; Lin, W.; Deng, C.; Zha, H.; Yang, X. A short survey of recent advances in graph matching. In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval, New York, NY, USA, 6–9 June 2016; pp. 167–174.
4. Yadav, S.K., Matching and Covering. In *Advanced Graph Theory*; Springer International Publishing: Cham, Switzerland, 2023; pp. 141–170.
5. Berge, C. *The Theory of Graphs and Its Applications*; Greenwood Press: Westport, CT, USA, 1982.
6. Avis, D. A survey of heuristics for the weighted matching problem. *Networks* **1983**, *13*, 475–493. [CrossRef]
7. Bunke, H. Recent developments in graph matching. In Proceedings of the 15th International Conference on Pattern Recognition. ICPR-2000, Barcelona, Spain, 3–7 September 2000; Volume 2, pp. 117–124.
8. Goddard, W.; Hedetniemi, S.M.; Hedetniemi, S.T.; Laskar, R. Generalized subgraph-restricted matchings in graphs. *Discret. Math.* **2005**, *293*, 129–138. [CrossRef]
9. Panda, B.; Pradhan, D. Acyclic matchings in subclasses of bipartite graphs. *Discret. Math. Algorithms Appl.* **2012**, *4*, 1250050. [CrossRef]
10. Baste, J.; Rautenbach, D. Degenerate matchings and edge colorings. *Discret. Appl. Math.* **2018**, *239*, 38–44. [CrossRef]
11. Chaudhary, J.; Mishra, S.; Panda, B. On the complexity of minimum maximal acyclic matchings. *J. Comb. Optim.* **2024**, *48*, 10. [CrossRef]
12. Chaudhary, J.; Mishra, S.; Panda, B. On the Complexity of Minimum Maximal Acyclic Matchings. In Proceedings of the Computing and Combinatorics: 28th International Conference, COCOON 2022, Shenzhen, China, 22–24 October 2022; Springer: Berlin/Heidelberg, Germany, 2023; pp. 106–117.
13. Panda, B.; Chaudhary, J. Acyclic matching in some subclasses of graphs. *Theor. Comput. Sci.* **2023**, *943*, 36–49. [CrossRef]
14. Panda, B.S.; Chaudhary, J. Acyclic matching in some subclasses of graphs. In *Proceedings of the International Workshop on Combinatorial Algorithms*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 409–421.
15. Fürst, M.; Rautenbach, D. On some hard and some tractable cases of the maximum acyclic matching problem. *Ann. Oper. Res.* **2019**, *279*, 291–300. [CrossRef]
16. Chaudhary, J.; Mishra, S.; Panda, B. Minimum maximal acyclic matching in proper interval graphs. *Discret. Appl. Math.* **2025**, *360*, 414–427. [CrossRef]
17. Hajebi, S.; Javadi, R. On the parameterized complexity of the acyclic matching problem. *Theor. Comput. Sci.* **2023**, *958*, 113862. [CrossRef]
18. Chaudhary, J.; Zehavi, M. Parameterized results on acyclic matchings with implications for related problems. *J. Comput. Syst. Sci.* **2024**, *148*, 103599. [CrossRef]

19. Fugacci, U.; Iuricich, F.; De Floriani, L. Efficient computation of simplicial homology through acyclic matching. In Proceedings of the 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 22–25 September 2014; pp. 587–593.
20. Saffren, S.; Angel, D. Acyclic matching on hypercube networks. In Proceedings of the 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), Kannur, India, 11–12 August 2022; pp. 681–685.
21. Fürst, M.; Rautenbach, D. A lower bound on the acyclic matching number of subcubic graphs. *Discret. Math.* **2018**, *341*, 2353–2358. [CrossRef]
22. Saffren, S.; Angel, D. Utilizing Acyclic Matching for Enhancing Current flow in Diamond Ladder Structures. In Proceedings of the 2023 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Jaipur, India, 17–20 December 2023; pp. 802–805.
23. Chaudhary, J.; Panda, B.S. On the complexity of minimum maximal uniquely restricted matching. *Theor. Comput. Sci.* **2021**, *882*, 15–28. [CrossRef]
24. Baste, J.; Fürst, M.; Rautenbach, D. Approximating maximum acyclic matchings by greedy and local search strategies. In Proceedings of the Computing and Combinatorics: 26th International Conference, COCOON 2020, Atlanta, GA, USA, 29–31 August 2020; Proceedings 26; Springer: Cham, Switzerland, 2020; pp. 542–553.
25. Fürst, M. Restricted Matchings. Ph.D. Thesis, Universität Ulm, Ulm, Germany, 2019.
26. Baste, J.; Fürst, M.; Rautenbach, D. Acyclic matchings in graphs of bounded maximum degree. *Discret. Math.* **2022**, *345*, 112885. [CrossRef]

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.