On Persistency and Maximum Cardinality Matching in Bipartite Graphs

Sofiat Olaosebikan[1], Frantisek Hajnovic[2], Nancy Ann Neudauer[3]

[1]African Institute for Mathematical Sciences AIMS-Ghana.
[2]Department of Computer Science, Comenius University, Bratislava Slovakia.
[3] Department of Mathematics and Computer Science, Pacific University, Forest Grove, OR 97116, USA
E-mail: sofiat@aims.edu.gh, ferohajnovic@gmail.com, nancy@pacificu.edu

January 22, 2017

Abstract

In maximum cardinality matching problems, a question of interest is which edges belong to all maximum matchings, no maximum matching and some but not all maximum matchings. Given a bipartite graph $G(X, Y, E)$ and any maximum matching in G, we describe an algorithm to answer this question in $O(n+m)$ time. This algorithm improves upon that of Tassa [1], which although with the same time complexity does not separate edges that belong to all maximum matchings from edges that belong to some but not all maximum matchings. Our algorithm also improves upon that of Costa [2], which solves the same problem in $O(nm)$ time, where n is the number of vertices and m is the number of edges. We further match the underlying ideas of our linear time algorithm to a method proposed by Régin [3] to filter constraints difference in Constraint Satisfaction Problems. Finally, we summarize the algorithms in pseudocodes and establish connections with other related work. .

Keywords: Bipartite graphs; Edge persistency; Maximum matching

1 Introduction

Many real world problems can be modelled as maximum cardinality matching problems in a bipartite graph. For example, job-applicant recruitment, student-project allocation and scheduling problems [4]. One of the questions we are often interested in is which edges belong to all maximum matchings, no maximum matching and some but not all maximum matchings. This property on the edges is known as *persistency*. It was first introduced for the maximum cardinality matching problem in bipartite graphs by Costa [2].

The following, as described in [4], is one of the motivations for this problem. "Suppose a pharmaceutical firm wishes to test p antibiotics on q volunteer subjects. However, prelimary screening has shown that certain subjects are allergic to certain of the drugs". This problem instance induces a bipartite graph with set of volunteers and set of drugs as the two color classes such that there is an edge between a volunteer and a drug if and only if the volunteer is not allergic to the drug. We note that some of those edges may not belong to any maximum matching. Thus, the pharmaceutical firm has an interest in identifying those edges beforehand and not offer them to any volunteer. In addition, with the knowledge of edges that belong to some but not all maximum matchings, the firm will have no hesitation in providing drug swap opportunities to some set of volunteers.

In 1994, Costa [2] presented an algorithm to make three-partition of the edge set of a bipartite graph $G(X, Y, E)$. The algorithm partitions E into three sets: E_0, the set of edges that belong to

no maximum matching (0-persistent edges); E_w, the set of edges that belong to some but not all maximum matching (weakly persistent edges); and E_1, the set of edges that belong to all maximum matchings (1-persistent edges). The complexity of the algorithm is $O(nm)$ time, with $|X \cup Y| = n$ and $|E| = m$.

Henceforth, we refer to the problem of making a three-partition of the edge set of any bipartite graph by persistency as ESP-3. It is worth noting that the ESP-3 problem can be generalised to other practical matching scenarios different from the one described above. Hence, the need to analyse and implement an efficient algorithm. In this paper, we describe a linear time algorithm to solve the ESP-3 problem, we provide clear pseudocodes and a comprehensive proof on which our improvement is based. In addition, we survey known approaches for solving other variants.

The idea which led to the linear time algorithm for the ESP-3 problem was a modification to some of the results presented in [2]. Thus, this linear time algorithm can be seen as a comprehension of the results in [2], in addition to a method which was used in [3] to filter costraints of difference in Constraint Satisfaction Problems (CSPs).

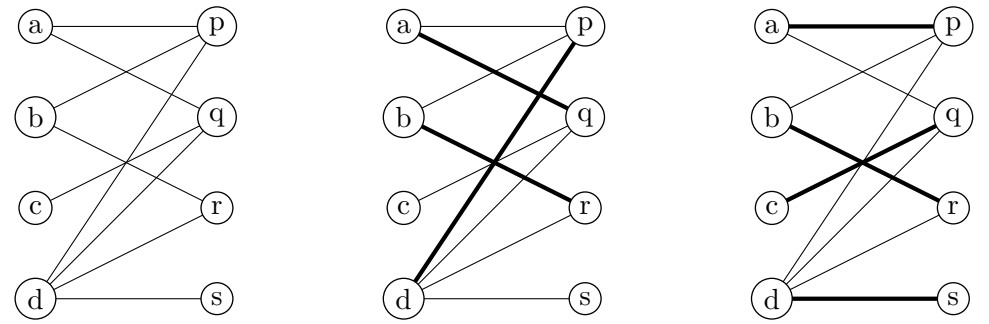
In 2012, Tassa [1] presented an algorithm to solve a variant of the ESP-3 problem. The algorithm finds all maximally-matchable edges (that is, edges that belong to some maximum matching) in G. In his case, the edge set is partitioned only into two sets: edges that belong to no maximum matching and edges that belong to some maximum matching. The algorithm runs in $O(n+m)$ time. One of the drawbacks of this algorithm is it cannot be used in a situation where knowledge of edges that belong to all maximum matchings of G is needed.

To implement the algorithms presented in [1] and [2], a maximum matching of the graph is needed. This can be obtained using the Hopcroft-Karp algorithm [5] which offers the best known worst-case performance with a runtime of $O(n^{1/2}m)$. We have added this algorithm to the Python library [8]. A better alternative is an algorithm by Alt et al [6] which finds a maximum matching in a bipartite graph in $O\left(n^{3/2}\sqrt{\frac{m}{\log n}}\right)$ time. It is an improvement by a factor of $\sqrt{log(n)}$, particularly for dense bipartite graphs.

In the next section, we give basic definitions and preliminaries. In section 3, we explain the procedure behind the algorithm presented by Costa in [2] to solve the ESP-3 problem. We provide a survey of algorithms needed, give their pseudocode, and explicitly tie them together, which is not done elsewhere. We present an algorithm with improved runtime in section 4. Section 5 gives a survey of other related work.

2 Definitions and Preliminaries

Let $G(X, Y, E)$ be an undirected bipartite graph with vertex sets X and Y and edge set $E \subseteq X \times Y$. A *matching* M of G is a collection of edges, subset of the edge set E, that share no common vertex. The *cardinality* of the matching $|M|$ is the number of edges it contains. M is *maximal* if it is not a proper subset of any other matching of G, that is, if any edge not in M is added to it, it is no longer a matching. M is *maximum* if there is no other matching of the graph with higher cardinality. An *edge is matched* if it is contained in a matching of the graph; otherwise it is *unmatched*. A *vertex is matched* if it is the endpoint of a matched edge; otherwise it is *exposed*. M is *perfect* if all the vertices of G are matched; otherwise it is *imperfect*. Figure 1 illustrates some of these definitions. A good reference for matching is Lovász and Plummer's *Matching Theory* [4].



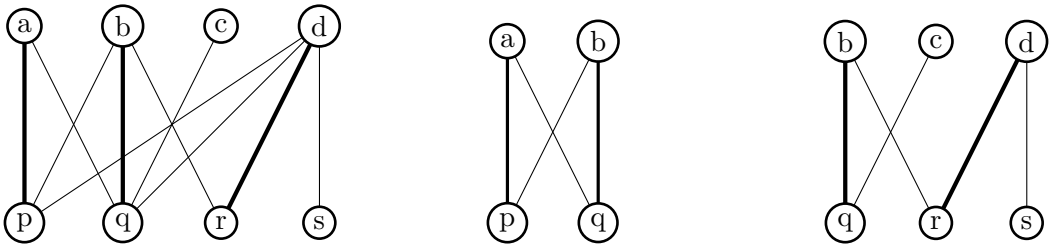
(a) Bipartite graph $G(X, Y, E)$

(b) Maximal and imperfect matching of $G(X, Y, E)$

(c) Maximum and perfect matching of $G(X, Y, E)$

Figure 1: A bipartite graph $G(X, Y, E)$ with a perfect and imperfect matching. The matched edges are the thick lines. In (b), edge $\{b, r\}$ and vertex b is matched while edge $\{c, q\}$ (vertex c) is unmatched (exposed).

To understand the algorithm presented in [2] for solving the ESP-3 problem, we need to define alternating paths, alternating cycles and augmenting paths. A path P_n in G is an *M-alternating path* if its edges are alternately in M and $E - M$. A cycle C_n in G is an *M-alternating cycle* if its edges are alternately in M and $E - M$. An M-alternating path whose first and last vertex are exposed in M is an *M-augmenting path.* Figure 2 illustrates these concepts.



(a) Matching M of a bipartite graph $G(X, Y, E)$

(b) M-alternating cycle

(c) M-alternating and M-augmenting path

Figure 2: A bipartite graph $G(X, Y, E)$ with a matching M.

The following theorem uses the notion of augmenting path to characterize when a matching is maximum.

Theorem 1 (Berge, 1957)**.** *A matching M in G is maximum if and only if G contains no M-augmenting path.*

Next, we present some useful results for the existing algorithm described in [2] for solving the ESP-3 problem.

3 An Algorithm for the ESP-3 problem

This section describes the algorithmic procedure presented by Costa for solving the ESP-3 problem. We briefly explain the approach behind the algorithm and provide a clear psuedocode. For proofs, readers should refer to [2]. Given any bipartite graph $G(X, Y, E)$, the approach assumes that we know one maximum matching M of G and uses some persistency properties proposed by Naddef [9], Dulmage and Mendelsohn [7], and Gallai and Edmonds [4]. Since a maximum matching M of G could either be perfect or imperfect, we first consider the case where M is imperfect.

3.1 Bipartite graph with an imperfect maximum matching

To solve the ESP-3 problem in a bipartite graph with an imperfect maximum matching, first, we partition the vertices of G into the Dulmage and Mendelsohn sets. We describe the sets and the procedure to obtain them.

The following sets are called the Dulmage and Mendelsohn sets [2].

- A_x - the set of vertices in X which are exposed in at least one maximum matching of G.
- A_y - the set of vertices in Y which are exposed in at least one maximum matching of G.
- B_x - the set of vertices in X that are not in A_x but adjacent to at least one vertex in A_y.
- B_y - the set of vertices in Y that are not in A_y but adjacent to at least one vertex in A_x.
- C_x - the set of vertices in X that are not in A_x and B_x, that is, $X - A_x - B_x$.
- C_y - the set of vertices in $Y - A_y - B_y$.

The following is a general result given by Gallai and Edmonds (see [4] for a proof).

Theorem 2. *If $G(X, Y, E)$ is a bipartite graph, and $A_x, A_y, B_x, B_y, C_x, C_y$ are as described above, then*

(1) $N(A_x) = B_y$ and $N(B_x) = A_y$;

(2) The subgraph $G(C_x \cup C_y)$ induced by C_x and C_y has a perfect matching, hence $|C_x| = |C_y|$;

(3) Every maximum matching of G consists of a perfect matching of $G(C_x \cup C_y)$, a maximum matching of A_x into B_y and a maximum matching of B_x into A_y.

In the psuedocode below, we describe a method to partition the vertices of G into the Dulmage and Mendelsohn sets, as proposed in [2].

Algorithm 1 PARTITION VERTICES OF $G(X, Y, E)$, GIVEN M.

Input: $G(X, Y, E)$, M
Output: L^+, L^* and L^u
  $L^+, L^*, L^u \leftarrow \emptyset, \emptyset, \emptyset$
  update L^+(resp. L^*) with exposed vertices in X (resp. Y)
  while there are vertices to label **do**
    update L^+ (resp. L^*) with all unlabelled vertices in Y (resp. X) adjacent to a labelled vertex in X (resp. Y) by an unmatched edge.
    update L^+ (resp. L^*) with all unlabelled vertices in X (resp. Y) adjacent to a labelled vertex in Y (resp. X) by a matched edge.
  end while
  $L^u \leftarrow V - L^+ - L^*$

The sets L^+ and L^* contain the vertices of G labelled $+$ and $*$ respectively while the set L^u contains the unlabelled vertices of G. We observe that the sets L^+, L^* and L^u form a partition of $X \cup Y$, since no vertex can be labelled $+$ and $*$, as this would lead to the existence of an augmenting path in the matching, thus the matching would not be maximum, as stated in Theorem 1. It was shown in [2] that the Dulmage and Mendelsohn sets are as follows:

$$\begin{aligned} A_x &= L^+ \cap X, \quad B_x = L^* \cap X, \quad C_x = L^u \cap X; \\ A_y &= L^* \cap Y, \quad B_y = L^+ \cap Y, \quad C_y = L^u \cap Y. \end{aligned}$$

We also note that the vertex sets A_x, B_x and C_x are pairwise disjoint, as are the sets A_y, B_y and C_y. The following theorem and its proof appear in [2]. It is key to classifying some edges of $G(X, Y, E)$ by persistency.

Theorem 3. *Let $G(X, Y, E)$ be a bipartite graph, M a maximum matching in G, and $A_x, A_y, B_x, B_y, C_x, C_y$ as defined above. Then*

- *edges in $B_x \times B_y$, $B_x \times C_y$ and $C_x \times B_y$ belong to the set E_0 (0-persistent edges);*
- *edges in $A_x \times B_y$ and $B_x \times A_y$ belong to the set E_w (weakly persistent edges).*

After implementing Algorithm 1 on a bipartite graph with an imperfect maximum matching, we have made a first decomposition of its vertex set. Using the output of this algorithm, we partition the vertices into the Dulmage and Mendelsohn sets, and all the edges with at least one labelled endpoint will be classified in either the set E_0 or the set E_w. In what follows, we present a clear pseudo code of the algorithm to solve the ESP-3 problem in a bipartite graph with an imperfect maximum matching.

Algorithm 2 PARTITION EDGES OF G WITH AN IMPERFECT MAXIMUM MATCHING M INTO THE SETS E_0, E_w AND E_1.

Input: $G(X, Y, E)$ and M
Output: E_0, E_w and E_1

$E_0, E_w, E_1 \leftarrow \emptyset, \emptyset, \emptyset$
$L^+, L^*, L^u =$ invoke Algorithm 1 on $G(X, Y, E)$
$A_x, B_x, C_x \leftarrow L^+ \cap X, L^* \cap X, L^u \cap X$
$A_y, B_y, C_y \leftarrow L^* \cap Y, L^+ \cap Y, L^u \cap Y$
for $\{u, v\} \in E$ **do**
    if $\{u, v\} \in B_x \times B_y$ or $\{u, v\} \in B_x \times C_y$ or $\{u, v\} \in C_x \times B_y$ **then**
        add $\{u, v\}$ to E_0
    else if $\{u, v\} \in A_x \times B_y$ or $\{u, v\} \in B_x \times A_y$ **then**
        add $\{u, v\}$ to E_w
    end if
end for
$E' \leftarrow E - E_0 - E_w$
$E'_0, E'_w, E'_1 \leftarrow$ invoke Algorithm 3 on $G'(C_x \cup C_y, E')$
update E_0, E_w and E_1 with E'_0, E'_w and E'_1 respectively

Analysis of Complexity: The algorithm initializes E_0, E_w and E_1 as empty sets. To determine the sets L^+ and L^*, the algorithm goes through all the edges of $G(X, Y, E)$ at most once (since a vertex is labelled at most once by $+$ or $*$), thus the procedure takes time $O(m)$. Once the Dulmage and Mendelsohn sets are obtained, the algorithm classifies some edges of $G(X, Y, E)$ within the for loop as elements of the sets E_0 and E_w. To do this, the algorithm checks the Dulmage and Mendelsohn set to which the endpoints of each edge belongs, which takes time $O(m)$. The overall time complexity is thus $O(m)$ in addition to the complexity for partitioning the edges of the subgraph $G'(C_x \cup C_y, E')$, which we describe in the next section.

According to Theorem 2, the subgraph $G'(C_x \cup C_y, E')$ induced by C_x, C_y and $E' = E - E_0 - E_w$ has a perfect matching. We observed that to solve the ESP-3 problem for a bipartite graph with a perfect matching, the algorithm described in [2] runs in $O(nm)$ time. In the next section, we present a linear time algorithm to solve the same problem.

4 A linear time Algorithm

In this section, we describe an improvement of the algorithm for solving the ESP-3 problem in a bipartite graph with a perfect matching.

4.1 Bipartite graph with a perfect maximum matching

Let $G(X, Y, E)$ be a bipartite graph and let M be any perfect matching in G. First, we construct a directed bipartite graph $G_d(X, Y, E_d)$ from G, after which we find all of its strongly connected

components. In what follows, we define the notion of a strongly connected component in a directed graph G.

Definition 1. *G is strongly connected if there is a directed path from each vertex in G to every other vertex. A strongly connected component of G is a maximal subgraph of G that is strongly connected.*

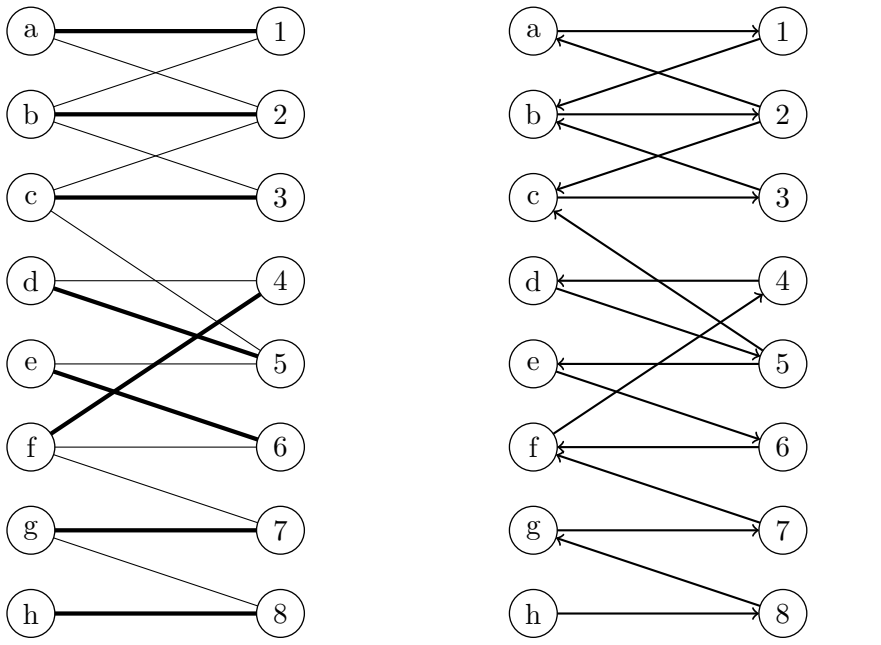
Next, we define the construction of the directed bipartite graph G_d.

4.1.1 Construction of directed bipartite graph

To construct $G_d(X, Y, E_d)$ from $G(X, Y, E)$, the sets of vertices X and Y remain the same. What follows is the definition of the edge set E_d.

$$E_d \quad = \quad \{(x, y) | (x \in X \text{ and } \{x, y\} \in M) \text{ or } (x \in Y \text{ and } \{x, y\} \in E - M)\}.$$

This edge set definition was adapted from Régin [3]. In the paper, Régin [3] was interested in removing every edge which belongs to no matching which covers X, a method which he used to design a filtering algorithm for constraints of difference in Constraint Satisfaction Problems. For an illustration of how to construct a new directed graph from the original bipartite graph, see Figure 3.



(a) Bipartite graph $G(X, Y, E)$ with a perfect matching M.

(b) Directed bipartite graph $G_d(X, Y, E_d)$

Figure 3: A bipartite graph $G(X, Y, E)$ with perfect matching M and the induced directed bipartite graph $G_d(X, Y, E_d)$.

For the graphs in Figure 3, each edge $\{x, y\} \in M$ forms the directed edge $(x, y) \in E_d$ while each unmatched edge $\{x', y'\} \in E - M$ forms the directed edge $(y', x') \in E_d$. Next we state a theorem given by Naddef [9] which provides information on which edges in Figure 3(a) are weakly persistent.

Theorem 4. *Let $G(X, Y, E)$ be a bipartite graph with a perfect matching M. An edge $\{x, y\} \in E$ is weakly persistent if and only if it belongs to an M-alternating cycle.*

In what follows, we present a theorem which claims that edges whose endpoints belong to the same strongly connected component belong to the same alternating cycle. Thus, by Theorem 4, those edges are weakly persistent.

Theorem 5. *Let $G(X, Y, E)$ be a bipartite graph with perfect matching M. Let $G_d(X, Y, E_d)$ be the induced directed bipartite graph constructed from $G(X, Y, E)$. An edge $\{x, y\} \in E$ belongs to an M-alternating cycle in G if and only if the endpoints x and y belong to the same strongly connected component in G_d.*

Proof.
($\Rightarrow$)

Let $C_k(G) = \{\{v_1, v_1'\}, \{v_1', v_2\}, \ldots, \{v_k, v_k'\}, \{v_k', v_{k+1} = v_1\}\}$ be an arbitrary M-alternating cycle in G. Since G is a bipartite graph, $C_k(G)$ is of even length and its edges alternate between M and $E - M$. Without loss of generality, suppose edges $\{v_i, v_i'\} \in M$, then edges $\{v_i', v_{i+1}\} \in E - M$, with $v_i \in X$ and $v_i' \in Y$, for $1 \leq i \leq k$ (otherwise, if edges $\{v_i, v_i'\} \in E - M$ and edges $\{v_i', v_{i+1}\} \in M$, with $v_i' \in X$ and $v_i \in Y$, we reverse or shift the indexing). By the construction of G_d, the matched edges $\{v_i, v_i'\} \in M$ induce the directed edges $(v_i, v_i') \in E_d$, while the unmatched edges $\{v_i', v_{i+1}\} \in E - M$ induce the directed edges $(v_i', v_{i+1}) \in E_d$. Hence we have constructed a directed cycle $C_k(G_d) = \{(v_1, v_1'), (v_1', v_2), \ldots, (v_k, v_k'), (v_k', v_{k+1} = v_1)\}$ and we observe that for every edge in $C_k(G)$, its endpoints belong to $C_k(G_d)$. Since $C_k(G_d)$ is a directed cycle, there is a directed path from each vertex to every other vertex in $C_k(G_d)$. This implies $C_k(G_d)$ is either a strongly connected component of G_d or is contained in a strongly connected component. Thus vertices in $C_k(G_d)$ belong to the same strongly connected component in G_d. Since $C_k(G)$ is an arbitrary M-alternating cycle, we conclude that if an edge belongs to an M-alternating cycle in G, then its endpoints belong to the same strongly connected component in G_d.

($\Leftarrow$)

Let $C_n(G_d) = ((v_1, v_1'), (v_1', v_2), \ldots, (v_n, v_n'), (v_n', v_{n+1} = v_1))$ be a strongly connected component in G_d. We note that every edge in $C_n(G_d)$ has its endpoints in $C_n(G_d)$. Since G_d is a bipartite graph, if $v_j \in X$, then $v_j' \in Y$, for $1 \leq j \leq n$, and vice versa. Suppose $v_j \in X$ and $v_j' \in Y$, then by the construction of G_d from G, the edges $(v_j, v_j') \in C_n(G_d)$ are precisely the matched edges in G and $(v_j', v_{j+1}) \in C_n(G_d)$ are the unmatched edges in G, that is, $\{v_j, v_j'\} \in M$ and $\{v_j', v_{j+1}\} \in E - M$. It follows that there is an M-alternating cycle $C_n(G) = \{\{v_1, v_1'\}, \{v_1', v_2\}, \ldots, \{v_n, v_n'\}, \{v_n', v_{n+1} = v_1\}\}$ whose edges alternate between M and $E - M$. In the case when $v_j' \in X$ and $v_j \in Y$, the argument follows the same procedure. Hence, we have shown that edges whose endpoints belong to the same strongly connected component in G_d belong to the same M-alternating cycle in G. □

Algorithm 3 presents the pseudocode for solving the ESP-3 problem in linear time, given a bipartite graph with perfect matching.

Algorithm 3 PARTITION EDGES OF $G(X, Y, E)$ WITH A PERFECT MATCHING M INTO THE SETS E_0, E_w and E_1.

Input: $G(X, Y, E)$ and M
Output: E_0, E_w and E_1

```
E_0, E_w, E_1 ← ∅, ∅, ∅
Construct G_d(X, Y, E_d) from G(X, Y, E)
SCCs ← Compute the strongly connected components of G_d
for each edge {x, y} in E do
    if vertex x and y belongs to the same strongly connected component in SCCs then
        add {x, y} to that strongly connected component
        add {x, y} to E_w
    end if
end for
for each edge {x, y} in M do
    if {x, y} does not belong to E_w then
        add {x, y} to E_1
    end if
end for
E_0 ← E − E_w − E_1
```

4.1.2 Overview of algorithm and analysis of complexity

Given a bipartite graph G with a known perfect matching M. Algorithm 3 first initializes the empty sets E_0, E_w and E_1. In line 2, to construct $G_d(X, Y, E_d)$ from $G(X, Y, E)$ by orienting all the edges in G takes at most $O(m)$ time. The strongly connected components of G_d are computed in line 3. Tarjan [11] described an algorithm to find all the strongly connected components of a directed graph in $O(n+m)$ time, where n is the number of vertices and m is the number of edges. Next, we add edges whose end points belong to the same strongly connected component to the set E_w (weakly persistent edges). Since E has at most m edges, the loop in line 4 runs in $O(m)$ time. In this loop, we also keep track of edges that belong to the same strongly connected component. In a practical setting, edges that belong to the same strongly connected component can be given the incentive to swap, without undermining the maximality of the underlying bipartite graph. The matched edges of G that do not belong to the set E_w are added to the set E_1 (1-persistent edges). Thus, line 10 also runs in $O(m)$ time. The 0-persistent edges of G are those in the set $E - E_w - E_1$.

Finally, given an undirected bipartite graph $G(X, Y, E)$, if the maximum matching M of G is imperfect, we run Algorithm 2. If M is a perfect matching, we run Algorithm 3. In any of these two cases, the overall time complexity to solve the ESP-3 problem in G is $O(n + m)$.

5 Related Work

Tassa [1] presented an algorithm for finding all the edges of a bipartite graph $G(X, Y, E)$ that are included in some maximum matching, that is, edges which are either weakly persistent or 1-persistent. With a known maximum matching, the time complexity of the algorithm is $O(n + m)$. We observed that his approach in the case when G has an imperfect maximum matching is different from Costa's approach. However, when G has a perfect matching, his approach is similar to that of the improvement we described, except that his algorithm partitions the edge set into two sets, instead of the three sets we seek.

In cases where there is one maximum matching, known without invoking any algorithm, the complexity of our algorithm and Tassa's algorithm is $O(n + m)$. However, our algorithm will be preferred over his in a situation where edges that extend to all maximum matchings of a bipartite graph is needed. In addition, the information our algorithm provides on the edges of G can be used to enumerate all the maximum matchings of G.

In [13], Carvalho and Cheriyan designed an algorithm for finding all maximally-matchable edges in general graphs in $O(nm)$ time. The matroid intersection algorithm presented in [12] can also be used to solve the ESP-3 problem on bipartite graphs. However, this produces an algorithm with complexity $O(n^5)$ time. This is clearly not efficient in our situation for bipartite graphs, since the algorithm we described here solves the same problem in linear time.

Although we arrived at the linear time algorithm described in Section 4 by focusing on improving the algorithm described in [2], we later found that the same idea was used by Régin in [3] to solve a different problem. Our goal was therefore to provide the linear time algorithm to solve the ESP-3 problem, along with concise psuedocodes, something which was not done elsewhere.

Acknowledgments

This work is the result of a master thesis carried out at the African Institute for Mathematical Sciences, AIMS-Ghana in 2015. We would like to appreciate AIMS for providing funding and support.

References

[1] T. Tassa Finding all maximally matchable edges in a bipartite graph, *Theoretical Computer Science* **423** (2012), 50 – 58.

[2] M. C. Costa, Persistency in maximum cardinality bipartite matchings, *Operations Research Letters* **15** (1994), 143 – 149.

[3] J. C. Régin, A filtering algorithm for constraints of difference in CSP's, *AAAI* **94** (1994), 362 – 367.

[4] L. Lovász and M.D. Plummer. *Matching Theory, Annals of Discrete Mathematics*. Elsevier Science Publishing B.V, 1986.

[5] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2:146 – 160, 1972.

[6] H. Alt, N. Blum, K.Mehlhorn, and M.Paul. Computing a Maximum Cardinality Matching in a Bipartite Graph in Time $O\left(n^{3/2}\sqrt{\frac{m}{\log n}}\right)$. *Information Processing Letters*, 37:237 – 240, 1991.

[7] A. L. Dulmage and N. S. Mendelsohn. (1958). Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10: 517 – 534, 1958.

[8] S. Olaosebikan. A Python library of the Hopcroft-Karp Algorithm. Python Package Index, `https://pypi.python.org/pypi/hopcroftkarp/1.2.4`

[9] D.J. Naddef. Rank of maximum matchings in a graph. *Math. Programming*, 22:52 – 70, 1982.

[10] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications*. Elsevier Science Publishing Co., Inc, 1976.

[11] R. Tarjan. Depth-First Search and Linear Graph Algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[12] K. Cechlárová and V. Lacko. Persistency in combinatorial optimization problems on matroids. *Discrete applied mathematics*, 110(2):121–132, 2001.

[13] M.H. de Carvalho and J. Cheriyan. An $O(nm)$ Algorithm for Ear Decomposition of Matching-Covered Graphs. *ACM Transaction on Algorithms (TALG)*, 1:324 – 337, 2005.