PROJECT REPORT

A study of Matchings in Graphs

Submitted in partial fulfilment of
the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

Submitted by

Chetan S
B070170CS

Nithin M Varma
B070025CS

Under the guidance of
Dr. K. Murali Krishnan

Department of Computer Science & Engineering
National Institute of Technology Calicut
Kerala - 673601
April 2011

Acknowledgements

We would like to sincerely thank our guide, Dr. K. Murali Krishnan (Assistant Professor, Dept. of Computer Science & Engineering, NIT Calicut), for his invaluable support and guidance towards this project. His motivation has always left us spellbound. We would also like to express our sincere thanks to Mrs. Lijiya A (Assistant Professor, Dept. of Computer Science & Engineering, NIT Calicut) for co-ordinating the project. We are grateful to Ms. Jasine Babu (PhD scholar, Indian Institute of Science, Bangalore) for going through the report and providing valuable suggestions. We also thank our friends and family for their help and support throughout.

Chetan S **Nithin M Varma**

B070170CS **B070025CS**

Declaration

We declare that the thesis entitled "A study of Matchings in Graphs" and the work presented in the thesis are both our own, and have been generated by us as the result of our own original research. We confirm that:

- this work was done wholly or mainly while in candidature for a Bachelors degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where we have consulted the published work of others, this is always clearly attributed;
- where we have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely our own work;
- We have acknowledged all main sources of help.

Place : Calicut
Date :

Chetan S
B070170CS

Nithin M Varma
B070025CS

Certificate

This is to certify that the project work entitled "**A study of Matchings in Graphs**", submitted by Chetan S (Roll No: B070170CS) and Nithin M Varma (Roll No: B070025CS), as the record of work carried out by them is accepted in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering at National Institute of Technology Calicut.

Guide
Dr. K. Murali Krishnan
Assistant Professor

Place : Calicut
Date :

Head of Department

Office Seal

Abstract

Matching in graphs and associated problems have been of great interest to theoretical computer scientists and mathematicians over decades. One of the celebrated results by Jack Edmonds showed that the matching problem in any general graphs yields itself to a polynomial time algorithm. Attempts have been made to analyse and bring down both the time and space complexities of this simple, yet intriguing problem.

In this report, we give a detailed description of the matching problem in graphs. We outline the techniques adopted to find the maximum matching and also analyse the complexity of the algorithms. Finally, we conclude with our observations on the complexity of the decision version of maximum matching i.e. deciding whether the given matching is maximum or not.

Contents

List of Figures

Chapter 1

Introduction

Jack Edmonds in his paper [4, 3] gave the first polynomial time algorithm to find a maximum matching in a general graph. This celebrated result, put the problem of finding the maximum matching in a general graph in **P**. Edmond's $O(n^4)$ algorithm for maximum matching was subsequently improved by Hopcroft et al. [5] and Micali et al. [9], independently, to a run-time of $O(\sqrt{n} \cdot m)$.

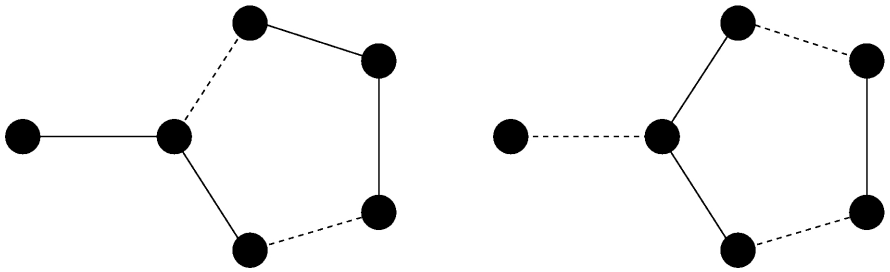
Efforts were also made on parallelising this problem, as the problem was not shown to be **P**-complete. The parallelisability of this problem comes from a theorem by Tutte [12] which says that a graph has a perfect matching if and only if a certain matrix of indeterminates, known as the Tutte matrix, is non-singular. The first algorithm to make use of this result was given by Lóvasz [8] who gave an **RNC** algorithm for the problem of deciding whether or not a graph has got a perfect matching. Borodin et al. also shows a similar result in [2]. Later, Karp et al. [6] showed that the problems of finding a perfect matching and a maximum matching in a graph are in $\mathbf{RNC}^3$. Their results were improved by Mulmulay, Vazirani and Vazirani to $\mathbf{RNC}^2$ in [10]. A long standing open problem in the area is to find out whether any or all of the above problems are in **NC**.

1.1 Preliminaries

A graph $G = (V, E)$ is a tuple consisting of a set of vertices V and a set of edges E. Let the sizes of the vertex set be $|V| = n$ and the edge set be $|E| = m$.

Definition 1. *(Matching) A matching of graph $G = (V, E)$ is a subset of the edges $M \subseteq E$ such that no two edges in M are adjacent.*

A vertex is said to be *covered* by matching M if any edge $e \in M$ is incident on it. A matching M is *perfect* if it covers all vertices. A perfect matching has a size of $\frac{n}{2}$. Figure 1.1 shows the maximal and maximum matching.



(a) Maximal matching (b) Maximum matching

Figure 1.1: Matching in graphs; dotted lines indicate matched edges

A matching M is *maximal* if the size of the matching ($|M|$) cannot be incremented by the addition of an edge $e \in E \backslash M$ (Figure 1.1(a)). A matching M is *maximum* if no other matching $M' \subseteq E$ has a higher cardinality ($|M'| > |M|$) (Figure 1.1(b)).

Definition 2. *(Augmenting path) A path P with alternating free and matched edges that begins and ends with free vertices.*

Figure 1.2 shows an augmenting path.

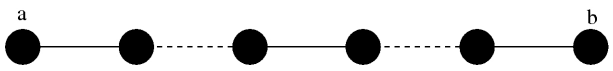


Figure 1.2: Augmenting path from free vertex a to b

Theorem 3. *(Berge) A matching M is maximum if and only if it has no augmenting path.*

Proof. We prove by considering both forward and backward directions separately.

($\Longrightarrow$) M is maximum $\Rightarrow$ M has no augmenting path.

Let P be the augmenting path in a graph $G = (V, E)$ with matching M. Flip the edges in path P (unmatched to matched and vice-versa). The resultant matching M' has higher cardinality than matching M.

$$|M'| = |(M \oplus P) \backslash E(P)| = |M| + 1$$

($\Longleftarrow$) No augmenting path $\Rightarrow$ M is maximum.

Let M' be a matching with $|M'| > |M|$. The symmetric difference $M' \oplus M$ yields a new graph with the following elements -

- Alternating paths of varying length.
- Cycles of even length.

Let $C_1, C_2, \ldots, C_k$ be the even cycles obtained in $M' \oplus M$. Then, we have -

$$|M \oplus C_1 \oplus C_2 \oplus \cdots \oplus C_k| = |M|$$

Since $|M'| > |M|$, there must be at least one alternating path in the symmetric difference $(M' \oplus M)$. This alternating path originates and terminates at vertices covered by matching M' thereby constituting an augmenting path in the original graph. □

Algorithm 1 MAXIMUM MATCHING(G)

```
1: M = φ
2: repeat
3:       M ← M ⊕ P
4: until augmenting path P exists
5: return M
```

Thus, as shown in Algorithm 1, the problem of finding a maximum matching reduces to a problem of finding augmenting paths efficiently.

Definition 4. *(Perfect Matching) A matching M in a graph G is said to be a perfect matching if and only if it is a maximum matching that matches every vertex in the graph.*

An obvious requirement for the existence of a perfect matching in a graph is that the number of vertices in it has to be even.

Chapter 2

Maximum Matching

2.1 Maximum Matching in Bipartite Graphs

Definition 5. *(Bipartite graph) A graph $G = (V, E)$ is bipartite if the set of vertices V can be partitioned into two disjoint sets, V_1 and V_2, such that there are no edges between any two vertices in the same partition V_i ($i \in \{1, 2\}$).*

The key property in a bipartite graph is that it does not contain cycles of odd length. This is straightforward to realise since the vertices forming the odd-cycle cannot be partitioned into two independent sets.

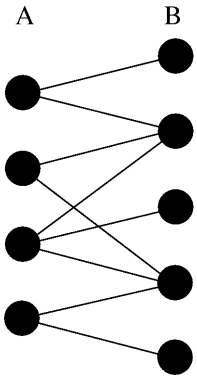


Figure 2.1: A bipartite graph with partitions A and B

A maximal matching of bipartite graphs can be found easily - adding edges to the matching until no more can be added. Also, a maximal matching M satisfies the property that $|M| \geq \frac{1}{2}|M'|$ where M' is the maximum matching of the bipartite graph. Thus, we can obtain a naive "2-factor approximation" to the maximum matching.

The maximum matching problem in bipartite graphs can be reduced to that of a flow problem. Let $G = (V, E)$ be a bipartite graph with partitions A and B. A directed graph $G'(V', E')$ is constructed from the original graph $G(V, E)$ by adding vertices and edges. The vertex set V' includes the vertices in V along with two special vertices - source vertex s and sink vertex t.

The edge set E' includes the edges in E directed from partition A to partition B. The source node s is connected to all the vertices in partition A and all the vertices in B are connected to the sink node t using directed edges. Each edge in E' has unit capacity.

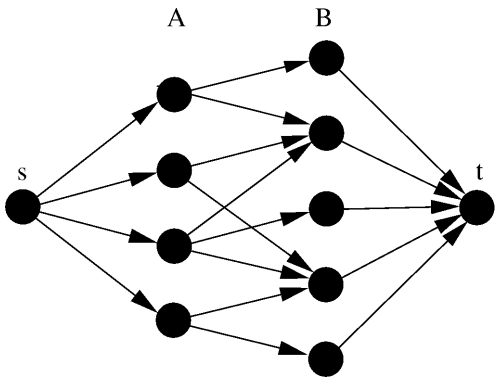


Figure 2.2: Flow graph for bipartite matching

A maximum flow f in the graph obeys the following properties -

1. All nodes in A have flows in at most one outgoing edge.
2. All nodes in B have flows in at most one incoming edge.

From these properties, we can conclude that the set of edges carrying the flow in f forms a maximum matching for the given graph G. Thus, solving a maximum flow in G' given a maximum matching for G. Any maximum flow algorithm like Ford-Fulkerson can be used to find the integral max-flows.

This algorithm runs in polynomial-time for any instance of bipartite graphs. The running time of Ford-Fulkerson is $O(m'F)$ where m' is the number of edges $|E'|$ and F is the maximum possible value of flow ($F < |V|$). Thus, the total run-time of the algorithm is $O(m' \cdot n) = O((m+n) \cdot n)$.

The maximum matching problem in bipartite graphs is also related to the vertex cover problem which is known to be **NP**-Complete.

Theorem 6. *(König-Egerváry) In a bipartite graph G, the maximum size of a matching is equal to the minimum size of a vertex cover.*

2.2 Maximum Matching in general graphs

The problem of finding a maximum matching depends on efficiently finding the augmenting paths. In odd-cycle free graphs or bipartite graphs, this is easy as we explore from each vertex only once while finding the augmenting path. However, this approach fails when there are odd-cycles in the graph.

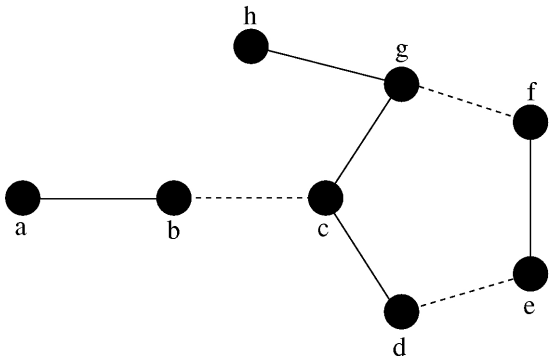


Figure 2.3: A matched graph with odd cycle

An alternating path from a free vertex a can reach vertex g either through a matched (fg) or unmatched edge (cg). A search for the shortest augmenting path from a will miss the longer augmenting path - a, b, c, d, e, f, g, h.

In 1965, Edmonds in his famous paper *Paths, trees and flowers* [4], found a way to overcome the problem of odd-cycles. The odd-cycles were handled by shrinking them into a super-vertex called 'blossom'.

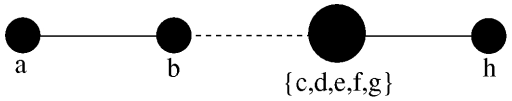


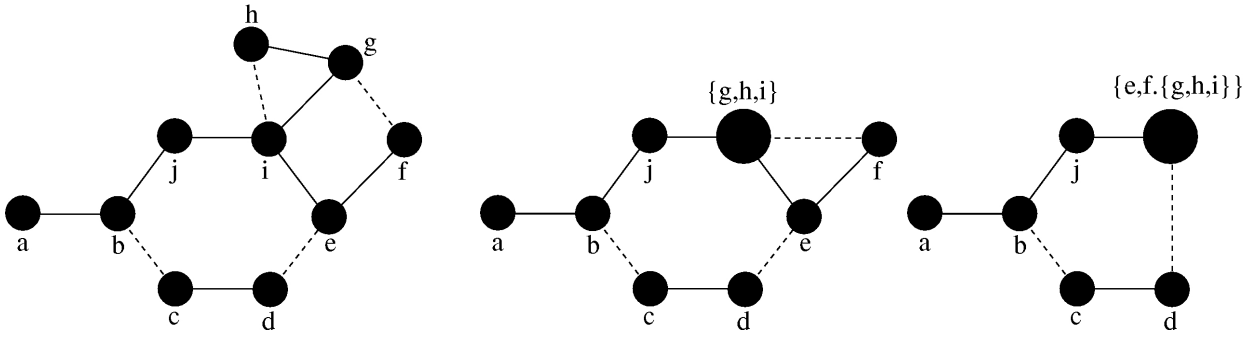
Figure 2.4: Odd cycle shrunk to a blossom - $\{c, d, e, f, g\}$

Definition 7. *(Flower, stem and blossom) A flower is the union of two alternating paths from a start vertex to a common vertex in the odd-cycle. A stem of the flower is the maximal common initial path. A blossom of the flower is the odd-cycle obtained by deleting the stem. The base of the blossom is defined as the common vertex of the blossom and the stem.*

In figure 2.3, a, b, c, d, e, f, g is the flower, a, b, c is the stem and c, d, e, f, g is the blossom.

The blossom does not hinder the search for an augmenting path. Any vertex in the blossom can be reached via a matched or unmatched edge and hence contracting the odd-cycle into a blossom helps in avoiding the various search paths.

The consolidation of the vertices into a blossom is done by contracting the edges whenever we encounter an odd-cycle. Once an augmenting path is found, the contractions are unrolled back to get an augmenting path in the original graph.

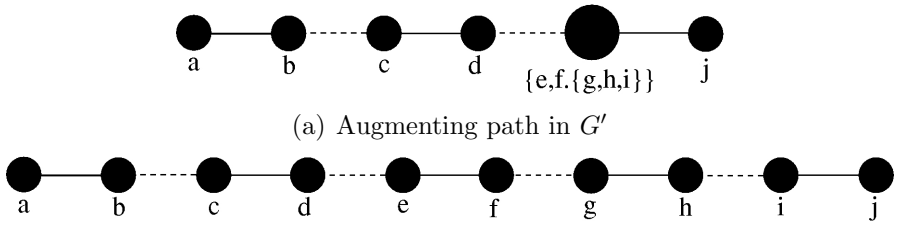


(a) Initial (b) Contraction 1 (c) Contraction 2

Figure 2.5: Odd cycles shrunk to a blossom - $\{e, f, \{g, h, i\}\}$

Example

The augmenting path can be determined as - $a, b, c, d, \{e, f, \{g, h, i\}\}, j$ from the figure 2.5(c).



(a) Augmenting path in G'

(b) Unrolled augmenting path in G

Figure 2.6: Finding the augmenting path in G

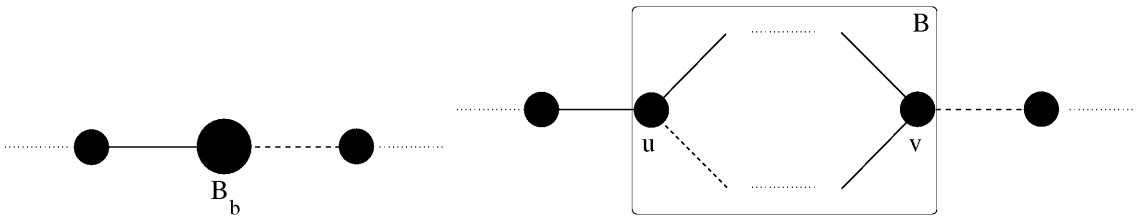
Lemma 8. *Let $G = (V, E)$ be a graph, M be any matching and B be a blossom. Let G' and M' be the new graph and matching obtained by shrinking $V(B)$ to the blossom B. Then, M is the maximum matching in G iff M' is the maximum matching in G'.*

Proof. By contradiction. We prove by considering both forward and backward directions separately.

($\Longrightarrow$) Suppose that M' is not a maximum matching in G'. Then, we prove that M is not a maximum matching in G.

Let Q be any augmenting path in G' whose length is k. Let B_b be the super vertex obtained from shrinking blossom B. If the vertices of the augmenting path Q do not touch the super vertex B_b, then unrolling the blossom will not affect the augmenting path in G. However, if Q touches B_b, then there are two vertices u and v in the unrolled blossom B which are unmatched and matched externally as shown in figure 2.7.

In the blossom B, the cardinality of unmatched edges $((V(B) + 1)/2)$ is one more than the cardinality of matched edges $((V(B) - 1)/2)$. There is only one vertex (v) in the blossom B having both unmatched edges. Hence, both the paths from u to v are alternating paths and Q can be extended to an augmenting path Q' in G. Thus, M is not a maximum matching in G.



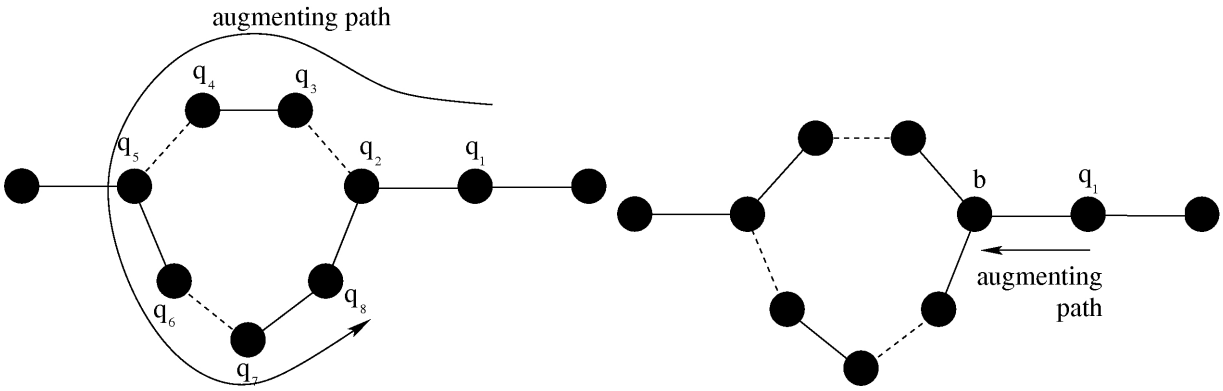
(a) Supervertex B_b in G' (b) Unrolled blossom B in G

Figure 2.7: Unrolling the blossom in G

($\Longleftarrow$) Suppose, M is not a maximum matching in G. Then, from Berge's theorem (Theorem 3), we have an augmenting path Q in G. Let $q_1, q_2, \ldots, q_k$ be the vertices of Q where $|Q| = k$. Then, there arises two cases -

- Both the end-points of path Q - q_1 and q_k - terminate outside the blossom.
- The end-point q_k terminates in the blossom.

In the first case, since both the end points terminate outside the blossom B, shrinking the graph will not affect the augmenting path. In the second case, given a matching M, we can always construct a matching N that switches the unmatched and matched edges in the alternating path $q_2, q_3, \ldots, q_k$ and $|M| = |N|$. This results in a graph where the base b of the blossom is unmatched. Hence, the new augmenting path is $q_1, q_2, \ldots, b$ and the problem is reduced to the first case. Figure 2.8 illustrates the proof.



(a) Augmenting path $q_1, q_2, \ldots, q_k$ (b) Augmenting path $q_1, q_2, \ldots, b$

Figure 2.8: Modifying the augmenting path

Thus, M' is not a maximum matching in G'. □

Chapter 3

Perfect Matching

3.1 Hall's Theorem for Bipartite Graphs

A necessary condition for perfect matching in a bipartite graph is the equal size of partitions A and B. Otherwise, the flow value (maximum matching) of the graph will be limited by the quantity $min(|A|, |B|) \leq |V|/2$. A necessary and sufficient condition for the perfect matching in bipartite graphs was given by Hall.

Theorem 9. *(Hall) A Bipartite graph $G = (V, E)$ has a perfect matching iff $|A| = |B|$ and for every subset $S \subseteq A$, the size of it's neighbors in B is atleast as large as $|S|$, i.e. $|N(S)| \geq |S|$.*

3.2 Tutte's Theorem for general graphs

3.2.1 Preliminaries

Before going into the statement of Tutte's theorem, we need to get ourselves acquainted with some preliminary ideas related to the same.

Definition 10. *(Connected Component) A connected component in an undirected graph is a maximal subset of its vertices, maximal w.r.t inclusion, such that every vertex in the subset is connected to every other vertex in it.*

Definition 11. *(Odd component) A connected component in a graph is an odd component if it has an odd number of vertices in it. The number of odd components in a graph G is denoted by $o(G)$.*

Lemma 12. *If M_1 and M_2 are two perfect matchings in a graph* G*, then the subgraph $M_1 \oplus M_2$ of* G *consists entirely of even cycles and isolated vertices.*

PROOF. The operation $\oplus$ stands for symmetric difference, i.e $M_1 \oplus M_2$ is a subgraph that consists of all the edges in the graph that belong to either of the matchings but not both. Since both M_1 and M_2 are perfect matchings, every vertex in the graph is saturated by both of them. If for a particular vertex, both the M_1 edge and the M_2 edge incident on it are same, that edge will not be included in the subgraph $M_1 \oplus M_2$. Hence the vertex will have degree 0 in $M_1 \oplus M_2$. If, on the other hand, the M_1 edge and the M_2 edge incident are not the same, then both of them will be included in $M_1 \oplus M_2$, and the vertex will have degree exactly 2. So, all the vertices in $M_1 \oplus M_2$ has degree wither 0 or 2. The degree 0 vertices are isolated vertices. Other than that, there are only degree 2 vertices. With a little thought, one can see that the remaining components can only be cycles. All such cycles will be of even cardinality. This is also trivial to see, as none of the vertices can have more than one edge belonging to the same matching incident on them. □

Tutte's theorem gives a necessary and sufficient condition for a perfect matching to exist in a given graph. The Tutte's condition is that for any subset S of the vertex set of G, number of odd components in G - S should be at most the cardinality of S. Tutte's theorem can be proved in a number of ways. In this report we'll present the proof as given by Lovasz[7]. The proof organisation and notations follow from [13].

3.2.2 Statement of the Theorem and Proof

Theorem 13. *[12] Any graph* G *has a perfect matching if and only if* G *satisfies $o(G-S) \leq |S|$ for every $S \subseteq V(G)$.*

PROOF. [13, 7]

Necessity

Consider a perfect matching M in G and take any subset S of $V(G)$. For every odd component in $G - S$, there exists at least one vertex which is connected via a matched edge to a vertex in S, for otherwise, M will not be a perfect matching. So, Tutte's condition holds for S and so does it for every subset of the vertex set.
Sufficiency

This is the tougher part of the proof. Before going into the proof, let us make and prove the following claim.

Claim 14. *Adding an edge to a graph which already satisfies Tutte's condition does not violate Tutte's condition.*

PROOF. This is because, addition of an edge can in no way increase the number of odd components of G - S, for any subset S considered. This becomes evident from the fact that two odd components getting connected by a new edge becomes an even component and an odd component and an even component getting connected via a new edge fuses into an odd

component. Finally, an edge added in between vertices of S as well as between a vertex in S and a vertex in G-S does not affect the number of odd components in G-S in any way. So, if G' is the graph obtained after adding an edge e to G, we have $o(G' - S) \leq o(G - S) \leq |S|$ for any $S \subseteq V(G) = V(G')$. □

Returning to the main proof, let us assume that there is a graph which satisfies Tutte's condition, has no perfect matching and the addition of any edge to it induces a perfect macthing. Let G be such an extremal graph. We obtain a contradiction by proving that G actually contains a perfect matching.

Define

$$U = \{v \in V(G) | d(v) = n - 1\}$$

We divide the proof into two cases based upon the structure of the connected components in G - U.

Case 1: G - U is a collection of disjoint cliques

The vertices in each of the even order cliques in G - U can be perfectly matched with the vertices in itself and all the odd order components have one extra vertex unmatched. Match this extra vertex with some vertex in U. There are enough vertices in U for this operation because G is assumed to satisfy Tutte's condition. It is easy to see that an even number of unmatched vertices remain in U which can be matched among themselves since $d(v) = n - 1$ for any $v \in U$. This gives a perfect matching in it.

Case 2: G - U is not a disjoint union of cliques

Since G - U is not a disjoint union of cliques, it is not tough to see that, there are at least two vertices v and w at a distance of at least 2 from each other. Therefore, we can assume that v and w are two vertices at a distance of exactly 2 (Why ?).Let $y \notin U$ be a common neighbour of v and w. In addition, $\exists x \notin U$ and x not a neighbor of y, since $y \notin U$. Since, from our earlier assumption, we know that, adding any edge to G creates a perfect matching in G, let M_1 and M_2 denote the matchings formed in $G + vw$ and $G + xy$, respectively. It is sufficient to show that $\exists$ a perfect matching in G avoiding both vw and xy. Consider $F = M_1 \oplus M_2$. Clearly, F does not contain any edge which is common to both matchings and the degree of every vertex in F is either 0 or 2. Put differently, F contains only even cycles and isolated vertices. Both vw and xy belongs to F. Let C be the cycle in F containing vw.

If xy also belongs to C, then the desired perfect matching which avoids these two edges consist of all those M_1 edges in the path from x to w or v (whichever is the nearer one) not through xy, the edge wy or vy as the case may be and all the M_2 edges from y to v or w (whichever is the nearer one) not through xy. To complete the perfect matching, add the remaining M_1 edges which are not in C, as well.

Now, if xy does not belong to C, the desired perfect matching consists of all the M_2 edges from C and all the M_1 edges not in C.

In both cases, we have shown a perfect matching in G. □

Perfect Matching is in NP∩co-NP

Tutte's Theorem gives a good characterisation for the existence of perfect matchings in a given graph G. A *Yes* certificate for the existence of perfect matchings is a set of edges which form a perfect matching of the graph. A Tutte set which violates the Tutte's condition would serve as an easily verifiable *No* certificate, indicating the absence of a perfect matching. This shows that the problem of finding a perfect matching is in **NP**∩co-**NP**.

Finding a perfect matching which would serve as a *Yes* certificate can be done in polynomial time by a run of the Edmond's algorithm[4, 3] which we already mentioned before. Exhibiting a *No* certificate, ie, a Tutte Set, is an **NP**-hard problem, as has been recently been shown in [1].

Chapter 4

Observations

Before going into the results, we'll review some basic complexity terminology which will be frequently referred to in the coming results and proofs.

Definition 15. NL *is the complexity class of all decision problems which can be solved by a nondeterministic* TM *using space logarithmic in the size of input.*

Definition 16. L *is the complexity class consisting of all the decision problems which are solvable by a deterministic* TM *using space logarithmic in the size of input.*

Definition 17. *A problem is said to be* **NL***-complete, if every problem in the class* **NL** *can be reduced into it via log-space reductions.*

Definition 18. *A reduction is said to be log-space, if it is computable by a deterministic* TM *using logarithmic space.*

The problems we have examined in our work are

- DECISION-MM : Given an undirected graph G along with a matching M, check whether the matching is maximum or not.
- AUG-PATH : Given an undirected graph G along with a matching M, check whether there exists an augmenting path between two free vertices s and t.

The key result of this report makes use of the following theorem due to Omer Reingold[11].

Theorem 19. USTCON $\in$ **L**

Here, USTCON is the problem of checking whether there exists a path between two vertices s and t in a given graph G.

4.1 Main Observations

The important observations which arose from this study are presented below.

Theorem 20. AUG-PATH $\in$ **NL**

Proof. Given a graph G with a matching M, the existence of an augmenting path between two free vertices s and t can be determined using the Algorithm 2.

Algorithm 2 AUGPATH(G, u, v)

```
1: parity := 0
2: Start from u
3: repeat
4:     if parity = 0 then
5:         Nondeterministically choose a neighbor w of u via an unmatched edge
6:     else
7:         Nondeterministically choose a neighbor w of u via a matched edge
8:     end if
9:     parity := 1 − parity
10:    u := w
11:    if u = v AND parity = 1 then
12:        return Yes
13:    end if
14: until all vertices have been examined
15: return No
```

□

Theorem 21. DECISION-MM $\in$ **NL**

Proof. Given a graph G with a matching M, checking whether it is maximum or not can be done using the Algorithm 3.

Algorithm 3 Maximum Matching(G)

```
1: for ∀(u, v) ∈ V × V do
2:     check := AUGPATH(G, u, v)
3:     if check = Yes then
4:         return No
5:     end if
6: end for
7: return Yes
```

The algorithm presented here invokes Algorithm 2 in its run. □

4.2 A direction towards *logspace* solution

Proposition 22. AUG-PATH $\in$ **L**

Attempt at Proof. To prove this result, we give a reduction from this problem to USTCONSET which we show to be in **L**.

Lemma 23. USTCONSET $\in$ **L**

Proof. Given a graph G, a vertex s and a set of vertices T, the problem of USTCONSET is to check whether there exists a path from vertex s to any of the vertices in T. This can be done in logspace using the logspace transducer for USTCON with slight modifications. The input to the machine is the three tuple $< G, s, T >$. This is written on the read-only input tape. For each vertex in $t \in T$, the machine checks for the existence of a path from s to t in successive iterations. The first time it finds a path, it halts by writing Yes on the output tape. Once all the vertices in T have been examined without any success, it halts by writing No on the output tape. Each of these iterations use only logarithmic space on the work tape. We can reuse the work tape for all iterations. Thus the machine uses no more than space logarithmic in the size of the input. □

Returning to the main proof, we will now show a logspace reduction from AUG-PATH to USTCONSET which will prove that AUG-PATH $\in$ **L**.

Given an instance $< G, s, t >$ of AUG-PATH, we output an instance $< G', s, T >$ of USTCONSET, such that an augmenting path exists between s and t in G iff there is a path connecting some vertex in T with s in G'. The working of the logspace transducer for the reduction is as illustrated by **??**.

Algorithm 4 Reduction(G, s, t)

```
1: if matching M is not proper then
2:     return Reject
3: end if
4: if either s or t is not a free vertex then
5:     return Reject
6: end if
7: E' := φ
8: for ∀(u, v) ∈ V × V such that u ≠ v do
9:     if ∃w such that ((u, w) ∉ M) and ((w, v) ∈ M) then
10:        E' := E' ∪ (u, v)
11:    end if
12: end for
13: G' := (V, E')
14: T := N(t)[1]
15: return < G', s, T >
```

It is easy to see that the reduction takes only space logarithmic in the size of input. For the iteration, we just need to keep two vertices u and v in the memory. In the execution of step

9, we need to check the specified conditions for each vertex in the graph. That again requires storing a single vertex at a time in memory.

The last part of the proof is to show that an augmenting path exists between s and t in G iff there is a path connecting some vertex in T with s in G'.

Claim 24. *There is a path in G' from s to some vertex $v \in T$ iff there is an augmenting path between s and t them in G.*

Proof. We prove both the forward and backward directions.

($\Longrightarrow$) Suppose there exists an augmenting path from s to t in G. The path starts and ends with unmatched edges. Let s be labelled u_1, t be labelled u_n and the intermediate vertices be labelled with u_2 to u_{n-1}. Clearly, edges (u_1, u_3),(u_3, u_5),... and (u_{n-3}, u_{n-1}) will be present in G' and hence there exists a path in G' connecting s and $u_{n-1} \in T$.

($\Longleftarrow$) Consider one such path from s to a vertex $v \in T$ in G'. Figure 4.1 shows such a path with s and v labelled p_1 and p_n. For the sake of clarity, we'll henceforth refer to edges in G'

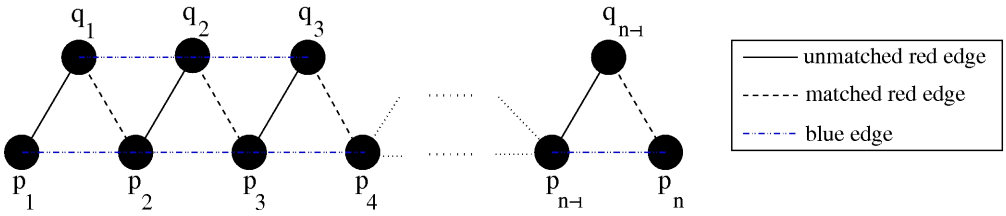


Figure 4.1: Path in G' labelled using blue edges

as blue edges and the edges in G as red edges. Since p_1 is a free vertex and p_1 is connected to p_2 via a blue edge, there should exist a vertex q_1 which is adjacent to both p_1 and p_2 through edges in G such that (p_1, q_1) is an unmatched edge and (q_1, p_2) is a matched edge. Similarly, for each $i \in \{2, 3, \ldots n-1\}$, there exists q_i that is adjacent to both p_i and p_{i+1} through edges in G such that (p_i, q_i) is an unmatched edge and (q_i, p_{i+1}) is a matched edge. All q_i's are unique because no vertex can have more than one matched edge going out of it. So, the path $(p_1, q_1, \ldots, p_i, q_i, p_{i+1}, \ldots, q_{n-1}, p_n)$ is an even length alternating path in G. Hence, $(p_1, q_1, \ldots, p_i, q_i, p_{i+1}, \ldots, q_{n-1}, p_n, t)$ is an augmenting path in G. □

However, the proof is incomplete as it does not consider the following case (as illustrated in figure 4.2) -

- A vertex with two adjacent unmatched edges and a matched edge.

From figure 4.2, in certain cases, the reduced graph G' includes a false alternating path (from a to c). Clearly, such cases should be avoided in the reduction. If this case can be patched, the above reduction proves that AUG-PATH $\in$ **L** □

[1]Neighbours of t in G

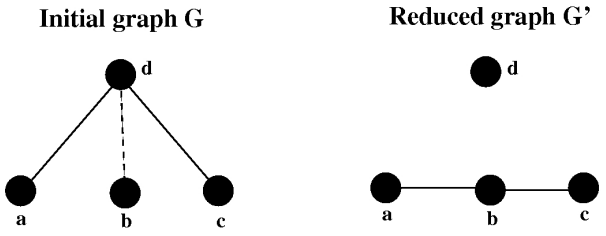


Figure 4.2: Pseudo alternating path $(a - b - c)$ in G'

If the above result can be satisfactorily proved for all cases, the following result comes as a consequence.

Corollary 25. *Claim 24* $\Longrightarrow$ DECISION-MM $\in \mathbf{L}$

Proof. This can be done in logspace using the logspace transducer for AUGPATH with slight modifications. The input to the machine is $< G >$. This is written on the read-only input tape. For each pair of free vertices in G, the machine checks for the existence of an augmenting path between them in successive iterations. The first time it finds a path, it halts by writing No on the output tape. Once all possible pairs have been examined without finding any augmenting path, it halts by writing Yes on the output tape. Each of these iterations use only logarithmic space on the work tape. We can reuse the work tape for all iterations. Thus the machine uses no more than space logarithmic in the size of the input. □

Chapter 5

Conclusions & Future Work

In this work, we have examined the space complexity of checking whether a given matching in a graph is maximum or not. We have shown that this problem belongs to the non-deterministic *logspace* complexity class (**NL**). We have also made an attempt at proving that the same problem is in **L**. But the proof fails to work for a particular case which we have illustrated in the report. We suggest patching the same as a potential problem to work on in the future.

Bibliography

[1] D. Bauer, H.J. Broersma, N. Kahl, A. Morgana, E. Schmeichel, and T. Surowiec. Tutte sets in graphs ii: The complexity of finding maximum tutte sets. *Discrete Applied Mathematics*, 155(10):1336–1343, May 2007.

[2] Allan Borodin, Joachim von zur Gathen, and John E. Hopcroft. Fast parallel matrix and gcd computations. *Information and Control*, 52(3):241–256, 1982.

[3] Jack Edmonds. Maximum matching and a polyhedron with $0, 1$ vertices. *J. of Res. the Nat. Bureau of Standards*, 69 B:125–130, 1965.

[4] Jack Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.

[5] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[6] R M Karp, E Upfal, and A Wigderson. Constructing a perfect matching is in random nc. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, STOC '85, pages 22–32, New York, NY, USA, 1985. ACM.

[7] L. Lovasz. Three short proofs in graph theory. *J. Comb. Theory*, 19:269–271, 1975.

[8] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

[9] Silvio Micali and Vijay V. Vazirani. An o($\sqrt{|V|}$|e|) algoithm for finding maximum matching in general graphs. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:17–27, 1980.

[10] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pages 345–354, New York, NY, USA, 1987. ACM.

[11] Omer Reingold. Undirected st-connectivity in log-space. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, STOC '05, pages 376–385, New York, NY, USA, 2005. ACM.

[12] W.T. Tutte. The factorisation of linear graphs. *J. Lond. Math. Soc.*, 22:107–111, 1947.

[13] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall of India Private Limited, 2nd edition, 2001.