Bipartite Graphs

MAD 3105: Discrete Mathematics II
Spring 2025

Adam Rumpf*

Some of the topics we will be covering in this course are not included in the main textbook [1]. As a result, we will occasionally rely on a free open-access textbook or on supplementary lecture notes as a reference. The following notes are primarily based on *Introduction to Graph Theory*, by Douglas West [2].

As a follow-up to our initial discussion of Eulerian graphs, we will take some time to discuss another important class of graphs: *bipartite* graphs. Bipartite graphs have many applications that we will discuss later in the course. As we did with Eulerian graphs, we will go through the process of deriving and proving a characterization for a bipartite graphs, and in doing so we will see another example of *TONCAS* ("The Obvious Necessary Condition is Also Sufficient") in Graph Theory.

1 Definitions

A **bipartite graph** is a graph whose vertex set can be partitioned into two independent sets, called its **partite sets**. All edges of a bipartite graph must begin in one partite set and end in the other (see Figure 1.1). A **bipartition** of a graph's vertex set is a partition into two independent sets.

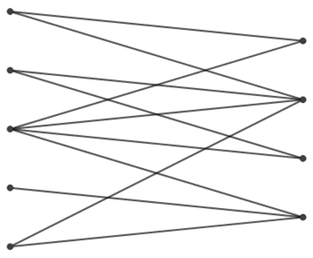

Figure 1.1: A bipartite graph. All edges must extend from one partite set (pictured on the left) to the other (pictured on the right).

A special case includes the **complete bipartite graph** on m and n vertices, denoted $K_{m,n}$, which is a bipartite graph with partite sets X and Y of size $|X| = m$ and $|Y| = n$, and with an edge between every possible pair of vertices with one in X and the other in Y (see Figure 1.2).

2 Applications

Bipartite graphs are typically use to represent problems that involve assigning agents from one set to agents from another. A prototypical example is: Suppose a company needs to hire several

*Florida Polytechnic University, Department of Applied Mathematics, arumpf@floridapoly.edu

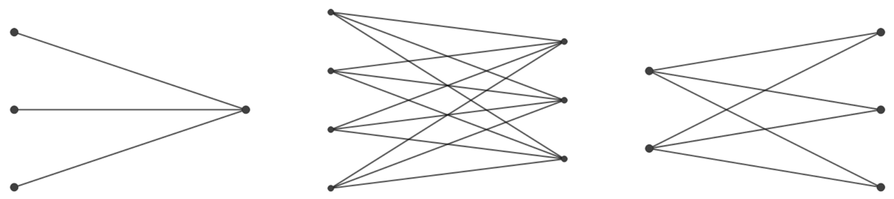

Figure 1.2: The complete bipartite graphs $K_{3,1}$, $K_{4,3}$, and $K_{2,3}$.

people to fill several open positions. Each candidate is qualified to perform some jobs but not others. The goal is to hire up to one person per job in order to fill all open positions. This can be modeled using a bipartite graph with one partite set whose vertices represent applicants and a second whose vertices represent jobs. An edge can be defined from each applicant to each job that they are qualified for. The goal of making the largest possible number of assignments corresponds to choosing the largest possible set of edges such that none share an endpoint (since an applicant can only be assigned to one job, and a job can only be filled by one applicant), which is a *maximum matching problem*.

As another example, suppose we have a set of suppliers, each with a limited supply of product, and a set of customers, each with their own demand, along with a unit shipping cost from each supplier to each customer. The goal is to determine how much product each supplier should send to each customer, with the objective of minimizing shipping costs, and while obeying the supply limitations and satisfying all customer demand. This is a *transportation problem*, and it is one of many *network flows* problems we will see later in the course.

3 Characterization

Attempting to prove that a graph is bipartite directly from the definition is theoretically challenging. Showing that a graph is bipartite requires showing that it has a valid bipartition, while showing that it is not requires showing that none of the exponentially-many possible bipartitions are valid. Fortunately there is a simple and computationally efficient characterization for a graph being bipartite, proved by Hungarian mathematician Dénes Kőnig in 1936.

Theorem 3.1 (Bipartite Graph Characterization). A graph is bipartite if and only if it contains no odd cycles.

Proof. This characterization is a biconditional statement, and requires showing *necessity* (the forward implication) and *sufficiency* (the reverse implication).

Necessity: Traversing a cycle in a bipartite graph requires traversing vertices that alternate between the two partite sets. Returning to the starting vertex requires alternating an even number of times, so an odd cycle cannot occur in a biparitite graph.

Sufficiency: Suppose G has no odd cycles. A graph is bipartite if and only if all its components are, so without loss of generality we may assume that G is connected. Choose an arbitrary vertex $u \in V$, and partition the vertices into two sets X and Y depending on whether their distance to u is even or odd. In particular, define

$$X = \{v \in V \mid \text{distance from } u \text{ to } v \text{ is even}\}$$
$$Y = \{v \in V \mid \text{distance from } u \text{ to } v \text{ is odd}\}$$

We claim that X and Y are both independent sets.

(a) Suppose to the contrary that there is an edge $\{v, w\} \in E$ for $v, w \in X$. Since v and w are both in X, there is an even path from u to v and from w to u. The even u, v-path, plus the edge $\{v, w\}$, plus the even w, u-path constitutes an odd cycle, which contradicts the assumption that G has no odd cycles.

(b) Suppose to the contrary that there is an edge $\{v, w\} \in E$ for $v, w \in Y$. Since v and w are both in Y, there is an odd path from u to v and from w to u. The odd u, v-path, plus the edge $\{v, w\}$, plus the odd w, u-path constitutes and odd cycle, which contradicts the assumption that G has no odd cycles.

In either case X and Y are independent sets that form a partition of V, thus G is bipartite. □

This is another famous example of a TONCAS result, in that the necessary condition is fairly easy and direct to show, but the fact that it is also sufficient is somewhat surprising and takes more work to demonstrate. However, with this characterization in place, there is a simple test for whether a graph is bipartite: if we find an odd cycle in a graph, then it cannot be bipartite; if not, then it is bipartite (see Figure 3.1).

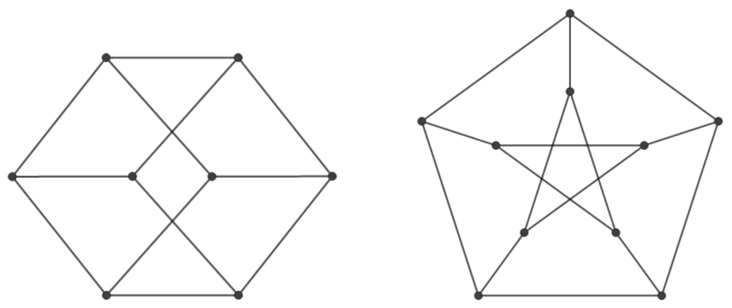

Figure 3.1: The graph to the left is bipartite. It contains no odd cycles, and a bipartition can be defined by alternating which set each adjacent vertex is placed in. The graph to the right is not bipartite, since it contains odd cycles (for example, the C_5 on the outer boundary).

There are efficient algorithms for finding odd cycles that make use of *search trees* (which we will see later in the course), thus testing whether a graph is bipartite is something that a computer can do very quickly and easily.

4 Bipartite Subgraphs

Not every graph is bipartite, but every graph contains a bipartite subgraph. A prototypical optimization problem is to try to find the largest possible (in terms of number of edges) subgraph of a particular type. Given the characterization from Theorem 3.1, we can make a trivial observation for the maximum number of edges in a bipartite subgraph.

Observation 4.1. The maximum number of edges in a bipartite subgraph of G is $|E|$ minus the minimum number of edges required to include one in every odd cycle of G.

This observation is based on the fact that a graph can be made bipartite by excluding enough edges to eliminate all odd cycles. The maximum number of edges in the subgraph is achived when all odd cylces have been eliminated by removing as few edges as possible. Actually computing this is usually computationally intractable, but there some lower bounds that can be achieved.

Theorem 4.2. Every simple graph G has a bipartite subgraph with at least $|E(G)|/2$ edges.

We can prove this claim algorithmically, by showing that the following algorithm produces the desired result:

1. Initialize X and Y as any arbitrary partition of V.
2. Repeat the following: Search each vertex $v \in V$ until finding one with more neighbors in its current partition than in the opposite one.
   (a) If such a vertex is found, move it to the opposite partition.
   (b) If no such vertex is found, halt.
3. Output the subgraph H with partite sets X and Y and with all edges in $E(G)$ between X and Y.

We claim that this algorithm is guaranteed to halt in finite time, and that the subgraph H it produces has at least $|E(G)|/2$ edges.

Proof. To show that the algorithm ends in finite time, note that a swap in step 2a is made if and only if doing so strictly increases the number of edges in H. Since each swap increases the number of edges in H by at least 1, and since G has finitely many edges, only finitely many swaps may occur.

To show that $|E(H)| \geq |E(G)|/2$ on termination, let $d_G(v)$ represent the degree of vertex $v \in V(G)$ in G, and $d_H(v)$ represent the degree of vertex $v \in V(H)$ in H. Within the main loop, a vertex v is moved to the opposite partite set if and only if it has more neighbors in its current set than in the opposite set. This means that, on termination, at least half the neighbors of v are in the opposite set, thus H includes at least half of the edges incident to v, so $d_H(v) \geq d_G(v)/2$ for all v. Then

$$\begin{aligned}
2|E(H)| &= \sum_{v \in V(H)} d_H(v) && \text{by the degree sum formula} \\
&\geq \sum_{v \in V(G)} d_G(v)/2 && \text{since } d_H(v) \geq d_G(v)/2 \text{ for all } v \text{, and } V(H) = V(G) \\
&= \frac{1}{2} \sum_{v \in V(G)} d_G(v) \\
&= \frac{1}{2} (2|E(G)|) && \text{by the degree sum formula} \\
&= |E(G)|
\end{aligned}$$

which can be rearranged to show that $|E(H)| \geq |E(G)|/2$. □

It should be noted that, although this algorithm guarantees a bipartite subgraph with *at least* half of the graph's edges, it is not guaranteed to find the bipartite subgraph with the *maximum* number of edges. Its stopping condition guarantees *local optimality*, not *global optimality.*

As an example, consider the graph displayed in Figure 4.1. The algorithm may halt with the bipartition $X = \{v_1, v_2, v_3, v_4\}$ and $Y = \{v_5, v_6, v_7, v_8\}$, which is locally optimal since no move of a single vertex from one partite set to the other would increase the number of edges in the bipartition. This bipartite subgraph includes 3 edges incident to each vertex, for a total of $|E(H)| = 12$ compared to $|E(G)| = 20$.

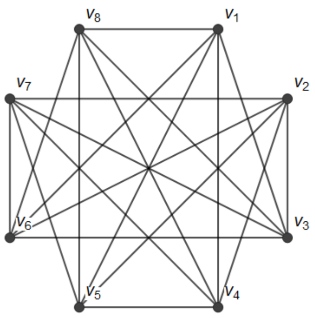

Figure 4.1: A graph that illustrates why the algorithm used to prove Theorem 4.2 does not generally guarantee the maximum number of edges.

However, there are bipartite subgraphs with more edges than this. The bipartition $X = \{v_1, v_2, v_7, v_8\}$ and $Y = \{v_3, v_4, v_5, v_6\}$ results in a bipartite subgraph with 4 edges incident to each vertex, for a total of $|E(H)| = 16$.

References

[1] R. Johnsonbaugh. *Discrete Mathematics.* Pearson, 8th edition, 2023. ISBN 9780137848577. URL `https://www.pearson.com/en-us/subject-catalog/p/discrete-mathematics/P200000006219`.

[2] D. B. West. *Introduction to Graph Theory.* Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2001. ISBN 0-13-014400-2.