Bipartite Matching

Eklavya Sharma

Abstract

This document describes an algorithm for perfect matching in bipartite graphs and the Hungarian Algorithm for min-cost perfect matching. Full proofs are not given, but the high-level idea is conveyed, so that it's intuitively clear what's going on in the algorithms.

1 Bipartite Graphs and Perfect Matching

Let $G = (L \cup R, E)$ be a bipartite graph, where L is the set of left vertices, R is the set of right vertices, and $E \subseteq L \times R$ is the set of edges. A subset $M \subseteq E$ of edges is called a *matching* in G if no two edges of M share an endpoint. If M is a matching and $|L| = |R| = |M|$, then M is called a *perfect matching*. See Fig. 1 for an example.

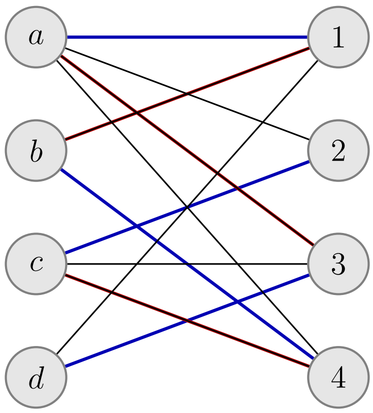

Figure 1: A bipartite graph. The blue edges form a perfect matching. The red edges form an imperfect matching.

Table 1: Graph of Fig. 1 in tabular format.

	1	2	3	4
a	✓	✓	✓	✓
b	✓	×	×	✓
c	×	✓	✓	✓
d	✓	×	✓	×

We can also represent bipartite graphs as a table. Each row of the table represents a left vertex and each column represents a right vertex. The cell (u, v) contains ✓ if an edge exists from u to v and $\times$ otherwise. See Table 1 for an example. A subset of edges forms a matching if for any two edges in that subset, their cells have different rows and different columns.

Here we will focus on two problems:

1. *Maximum cardinality matching*: Given a bipartite graph, find a matching M such that $|M|$ is maximized.

2. *Min-cost perfect matching*: Given a bipartite graph, and given the cost of edge edge, find a minimum-cost perfect matching, or report that no perfect matching exists.

2 Primal-Dual Technique

Sometimes, it's hard to solve a problem but easy to verify the solution. E.g., in the factoring problem, we are given a large natural non-prime number x, and we need to output two natural numbers y and z larger than 1 such that $x = yz$. Factoring the number $x = 2252989$ by hand is quite time-consuming. But if I tell you that the factors are $y = 1117$ and $z = 2017$, it's easy to verify that they are indeed the factors by simply multiplying these numbers to compute yz and then checking that x equals yz. Similarly, solving a sudoku puzzle is not easy, but verifying a solution is simple: just check that each row, column, and 3×3 box is a permutation of $\{1, 2, \ldots, 9\}$.

The ability to easily verify solutions to a problem can make life much easier. Imagine spending a lot of time trying to solve a problem, and having no way of easily checking whether you got the right answer. That would be bad, right? Also, verifiability helps us use *hit-and-trial* to solve the problem, i.e., repeatedly guess a solution and check if it's a correct solution.

For some problems, it's not obvious how to verify the solution. E.g., for linear programming, given a vector x, it's not obvious how to easily check whether x is an optimal solution to the LP. We can find the optimal objective value of the LP (e.g., using the simplex method) and then compare that to x's objective value to check whether x is optimal, but that would be very time-consuming.

However, suppose we want to solve both the primal and the dual LPs. Given two vectors x and y, it is easy to check whether they are optimal solutions to the primal and dual LPs, respectively, using the strong duality of linear programs: just check that x is feasible for the primal, y is feasible for the dual, and that x and y have the same objective value.

This technique of solving problems by simultaneously finding a primal and dual solution is called the *primal-dual technique*. We will use this technique in the algorithms in this document.

3 Maximum Cardinality Matching

3.1 Hit-and-Trial

Definition 1. *In a bipartite graph $G = (L \cup R, E)$, a set $S \subseteq L \cup R$ of vertices is a vertex cover if for every edge in E, at least one of its endpoints is in S.*

In the minimum cardinality vertex cover problem, we are given a bipartite graph and we need to find a vertex cover S such that $|S|$ is minimum. We can use linear programming duality to show that this problem is the dual of max cardinality matching. Hence, if M is a matching and S is a vertex cover, then $|M| \leq |S|$. Although this can be proved using weak LP duality, we will give a direct proof of this fact:

Lemma 1 (weak duality: matching vs vertex cover)**.** *Let $G = (L \cup R, E)$ be a bipartite graph. Let $M \subseteq E$ be a matching and $S \subseteq L \cup R$ be a vertex cover. Then $|M| \leq |L|$.*

Proof. For each edge in M, at least one of its endpoints lies in S. Since edges of M don't share endpoints (because it's a matching), we get that there must be at least $|M|$ vertices in S. □

Therefore, if we can find a matching M and a vertex cover S such that $|M| = |S|$, then M is a maximum cardinality matching (and S is a min vertex cover). See Fig. 2.

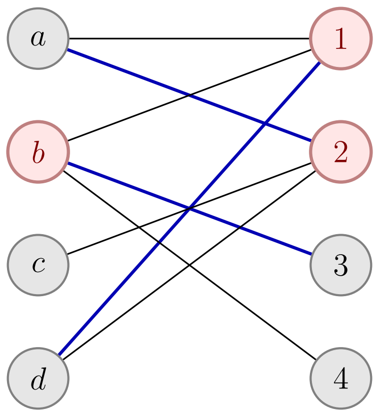

Figure 2: Blue edges give us a matching M. Red vertices give us a vertex cover S. We know that M is a maximum cardinality matching in this graph since $|M| = |S|$.

	1	2	3	4
a	✓	✓	×	×
b	✓	×	✓	✓
c	×	✓	×	×
d	✓	✓	×	×

Furthermore, this approach is guaranteed to work, i.e., if M is a maximum cardinality matching and S is a minimum vertex cover, then $|M| = |S|$. We cannot prove this using strong LP duality alone. We would also need to prove that optimal solutions are integral. We can prove it using the max-flow min-cut theorem (the proof is beyond the scope of this document).

3.2 Augmenting Path Algorithm

(TODO)

4 Min-Cost Perfect Matching

(TODO)