A PRIMAL-DUAL ALGORITHM FOR THE UNCONSTRAINED FRACTIONAL MATCHING PROBLEM

by

Bobby Chan

BSc., Simon Fraser University, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Fall 2009

APPROVAL

Name:	Bobby Chan
Degree:	Master of Science
Title of Thesis:	A Primal-Dual Algorithm for the Unconstrained Fractional Matching Problem

Examining Committee: Dr. Pavol Hell
Chair

Dr. Ramesh Krishnamurti, Senior Supervisor

Dr. Binay Bhattacharya, Supervisor

Dr. Daya Gaur, External Examiner,
Associate Professor of Math and Computer Science,
University of Lethbridge, Lethbridge, AB

Date Approved: Aug 4th, 2009

Abstract

In this thesis, we present a primal-dual algorithm for a generalization of the *Maximum Matching Problem*, a well known problem in graph theory. Although the proposed problem, called the unconstrained fractional matching problem, can be solved in polynomial time using established algorithms for linear programming, we provide a combinatorial algorithm with a running time complexity of $O(|V|^2|E|)$.

Keywords

factional matching, matching, charge problem, graph theory, primal-dual, linear programming, combinatorial algorithm

To mom, dad, grandma, and Rachel for your endless support and inpiration

To my saviour Jesus Christ, without whom nothing is possible.

"Where is the wise man? Where is the scholar? Where is the philosopher of this age? Has God not made foolish the wisdom of the world? For since in the wisdom of God the world through its wisdom did not know Him, God was pleased through the foolishness of what was preached to save those who believe."

— *1 Corinthians 1:20-21,* NIV

Acknowledgements

I must begin by thanking my senior supervisor Dr. Ramesh Krishnamurti for his endless help and support both personally and professionally. I am very grateful for his patience through all the problems I burdened him with along the way and for doing such a thorough job of reviewing this thesis and pointing out details which otherwise I would have overlooked.

One of the toughest tasks in grad school, I've been told, is to find a supervisory committee who is compatible with you on a personal level. I feel privileged to have worked with Dr. Daya Gaur, Dr. Binay Bhattacharya, and Dr. Ramesh Krishnamurti who are not only extremely knowledgeable but are among the nicest people I have ever met.

Professionally, the past three years have been the most enjoyable period of my life. I have discovered a love for teaching thanks to the encouragement of mentors such as Dr. Diana Cukierman and Dr. Tony Dixon whose confidence in me allowed me to grow as an instructor.

Finally, and most importantly, I would like to thank those closest to me. My mom, my dad, and my grandma, whose unconditional love, patience, and support has made it easy for me to succeed. My soon to be wife Rachel, whose faith stabilized me through the ups and downs of this academic venture. My saviour and redeemer Jesus Christ, without whom all the work done in this thesis would be meaningless.

Contents

List of Figures

Chapter 1

Introduction

In mathematics and computing science, a graph $G(V, E)$ is a set of *vertices* V, and a set of *edges* E comprised of pairs of vertices $(u, v) \in V$. Given an edge $e = (u, v) \in E$, vertices u and v are said to be *adjacent.* A *matching* in a graph is a subset of edges which have a disjoint set of adjacent vertices. Given a matching M in a graph, the size of the matching, denoted $|M|$, is the number of edges in M. The problem of finding the maximum matching for a given graph is well-known in graph theory. The challenge becomes significantly more difficult when weights and/or capacities assigned to the edges and vertices in G are not equal to 1. The basic matching problem, which has been well studied throughout the years, is one in which weights and capacities of both edges and vertices are equal to 1. In this thesis, we will present a literature survey on the work that has been done in maximum matching for general and bipartite graphs [HK73, Vaz94] where edge values can be both integers as well as fractions [EK72, Edm65b, BP89]. However, our motivation is to extend the work here and apply it to the *maximum charge problem* [KGGS06], a generalization of the maximum matching problem in which both edges and vertices can have capacities other than 1.

The maximum matching problem (as well as many other problems in graph theory) can be formulated as a mathematical program, called a *linear program*, with a linear objective function which is to be maximized and a set of linear inequalities to be satisfied. Such a linear program can be solved in polynomial time by interior point methods, as well as by the *simplex algorithm.* Even though the simplex algorithm does not necessarily terminate in polynomial time, it works well in practice. While simplex is able to solve all linear programming problems, it is much slower than most combinatorial algorithms for graph theory problems. Fortunately, for some linear programs, the *primal dual algorithm* [DLF56] provides a structure that allows us to design more efficient combinatorial methods.

The work presented here is submitted as part of the requirements of the Master's thesis and is organized into five chapters. We will begin with a short discussion of the maximum matching problem including representative variations, history, as well as running times. Next, the primal dual algorithm will be formally introduced. We will then use a primal-dual schema for a generalization of the maximum matching problem and provide a combinatorial algorithm with running time $O(|V|^2|E|)$ for the same. Finally, we evaluate the difficulties of this approach for the maximum charge problem as well as suggest possible research goals for the future.

Chapter 2

Matching

2.1 Introduction

Given a graph $G(V, E)$, a *matching* M is a subset of edges in G such that no two edges share a common vertex. The *Maximum Matching Problem*, a well known problem in graph theory, attempts to find a matching that contains as many edges as possible.

The Maximum Matching Problem comes in many variations and has various applications in networking and artificial intelligence. Variations in the problem include adding weights to edges [Kuh55], limiting the structure of the graph G [BP89], as well as adding capacities to both edges and vertices [KGGS06]. Because of its augmentative nature, algorithms to solve this problem are very closely related to solutions of the *Maximum Network Flow Problem* [AMO93].

2.2 0-1 Matching

Given a matching M in graph $G(V, E)$, a vertex $u \in V$ is said to be *saturated with respect to M* if there exists an edge incident to u in M; otherwise u is *unsaturated.* A path $P = \{u_1, e_1, u_2, e_2, ..., u_k, e_k, u_{k+1}\}$ is a sequence of vertices and edges such that each vertex u_i is connected to u_{i+1} through an edge $e_i \in E$ (for $i = 0...k$). A path P is *alternating with respect to M* if the edges $e_1, e_2, ..., e_k$ in P belong alternately to $E - M$ and M. An alternating path P is *augmenting with respect to M* if P begins and ends at unsaturated vertices and contains an odd number of edges. These definitions will be formalized in subsequent sections.

One of the most important theorems in matching theory is due to Berge in 1957 [Ber57].

Theorem 2.2.1. *(Berge's Theorem) Given a graph G, a matching M is maximum in G if and only if there exists no augmenting path with respect to M.*

Proof. Assume there is an augmenting path P with respect to M. It is trivial to see that a new matching $M^* = M \triangle P$ will increase the size of the matching $|M|$ by 1, where $\triangle$ denotes the symmetric difference of two sets. It is left to show that if a matching M is maximum, there will exist no augmenting path with respect to M. We will make the following assumptions:

(i) M is a matching in G that is *not maximum.*
(ii) M^* is a maximum matching.
(iii) Among all maximum matchings, M^* minimizes $|M^* - M|$. That is, M^* has the most edges in common with M.

Consider the set $S = M^* \triangle M$. There cannot exist paths of even length because of assumption (iii). So there are only odd length paths in S. Now, from all odd length paths in S, there must be at least one which has more edges in M^* than M since $|M^*| > |M|$. Hence, we have found an augmenting path with respect to M. $\square$

The maximum matching problem can be formulated as an *integer program.* More specifically, in the maximum matching problem, the variables corresponding to the edges can only take values 0 or 1. Such a formulation is called a *0-1 integer program.* The general version of the 0-1 integer program is an NP-complete decision problem [Kar72]. Given a graph $G(V, E)$, the integer program for the maximum integer matching problem is as follows:

$$\begin{array}{lll} \textit{maximize} & \sum_{e \in E} \pi_e & \\ \textit{subject to} & \sum_{e \in E(u)} \pi_e \leq 1 & \forall u \in V \\ & \pi_e = \{0, 1\} & \forall e \in E \end{array} \tag{IMP}$$

π_e is a variable corresponding to edge e for each edge $e \in E$. Whenever π_e equals 1, the corresponding edge e is in the matching. Because of its structure, the integer matching problem is not NP-complete even though it is a 0-1 integer program. We will see in subsequent chapters that relaxing the edge value constraints will allow us the flexibility to apply some interesting techniques to solve this problem.

2.2.1 Bipartite 0-1 Matching

A given graph $G(V, E)$ is *bipartite* if its vertices can be divided into two disjoint sets U_1 and U_2 ($U_1 \cup U_2 = V$) such that both U_1 and U_2 are independent sets. In other words, each edge $e \in E$ connects a vertex in U_1 with one in U_2. Maximum bipartite 0-1 matching is perhaps the least complicated form of matching because of its simple structure. One method to solve this is to first add a super source s, and a super sink t to the graph G, with infinite capacity edges connecting s (t) to all vertices in U_1 (U_2) while the capacities on all other edges are set to 1. Next, we apply a maximum flow algorithm to the modified graph. It may be shown that all the edges with flow equalling 1 are the edges in the matching. This algorithm runs in $O(|V|^3)$ time [KM74,MPKM78]. In 1973, using another approach, Hopcroft and Karp were able to improve the running time to $O(|E||V|^{\frac{1}{2}})$ by finding the maximal set of vertex disjoint minimum length augmenting paths using a modified breadth first search in a series of $\sqrt{|V|}$ *phases* [HK73]. This is still widely regarded as the fastest maximum bipartite matching algorithm although it has recently been shown that the running time can theoretically be improved to $O(|V|^{2.38})$ [MS04].

It is also worth noting that in 1931, König and Egerváry [Ege31,K$\ddot{3}$1] showed that any regular bipartite graph[1] has a perfect matching[2]. In addition, König's matching theorem states that given a bipartite graph, the size of the maximum matching is equivalent to the size of the *minimum vertex cover*[3] [Riz00].

2.2.2 General 0-1 Matching

While Berge's theorem was a breakthrough in maximum matching, it was still difficult to apply it directly to general graphs. The Hopcroft and Karp algorithm mentioned in the previous section depends on the fact that we will never encounter an odd length cycle while traversing a bipartite graph using alternating breadth first search. The existence of an odd length cycle within an augmenting path, also called a *blossom*[4], results in the modified breadth first search giving false positive as well as false negative results while searching

[1]one in which all vertices have the same degree

[2]Given a graph $G(V, E)$, a matching M is *perfect* if each vertex $v \in V$ is adjacent to an edge in M

[3]Given a graph $G(V, E)$, a *vertex cover* C is a subset of vertices in G that is adjacent to at least one edge in G. A vertex cover is minimum if no other covers have fewer vertices.

[4]Given a matching M of G, a *blossom* B is an alternating path with respect to M that forms a cycle. The vertex adjacent to two unsaturated vertices in B is called the *base* of the blossom (see figure 2.1)

for augmenting paths. In figure 2.1, we see the path represented by the arrows will be found by the modified BFS as an augmenting path even though one does not exist. In 1965, Edmonds devised a method to effectively "shrink" these blossoms if there exist no augmenting paths which use an edge within these blossoms. In the same paper, Edmonds suggested a polynomial time algorithm which runs in $O(|V|^4)$ time [Edm65b]. This result was eventually improved to $O(|V|^{\frac{5}{2}})$ by Vazirani in 1980 by once again finding the shortest augmenting path in a series of $\sqrt{|V|}$ phases [Vaz94].

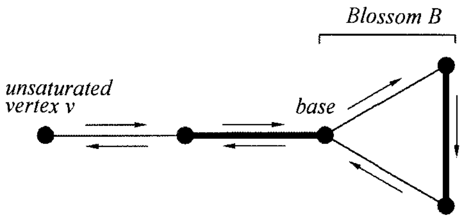


Figure 2.1: A blossom and a false positive augmenting path

2.3 Maximum Fractional Matching

As stated in section 2.2, the regular 0-1 matching problem can be represented using an integer program. A relaxed version of an integer program, called a *linear program* allows variables to be assigned fractional values between 0 and 1. For this reason, the relaxed version of the 0-1 matching problem is referred to as the *fractional matching problem* and can be formulated as follows (examples of LP formulations as well as optimal solutions are illustrated in appendix A):

$$\begin{array}{lll}
\textit{maximize} & \sum_{e\in E} \pi_e & \qquad\qquad \text{(FMP)} \\
\textit{subject to} & \sum_{e\in E(u)} \pi_e \leq 1 & \forall u \in V \\
 & \pi_e \leq 1 & \forall e \in E \\
 & \pi_e \geq 0 & \forall e \in E
\end{array}$$

We will note that a given edge value π_e can "fractionally" belong to a matching M. It is easy to see that, given an instance of the matching problem, the optimal solution to the

LP is greater than or equal to the optimal solution to the IP. In this paper, we will focus primarily on the fractional version of the matching problem.

Given a bipartite graph G an optimal solution to the fractional matching problem is equivalent to the optimal solution to the 0-1 matching problem. This is due to the fact that there are no odd length cycles in a bipartite graph and that the associated incidence matrix is unimodular [SU97].

In 1989, Bourjolly and Pulleyblank proved that given a general graph, the optimal solution to the fractional matching problem is half-integral. In other words, an optimal solution can be constructed such that $\pi_e \in \{0, \frac{1}{2}, 1\}$ for every edge e [BP89] and can be constructed in $O(|E||V|)$ time.

2.4 Maximum Charge Problem

The *maximum charge problem* is a generalization of the maximum fractional matching problem in which capacity constraints are assigned to both the edges and vertices [KGGS06]. Given a graph $G(V, E)$, for each vertex $u \in V$, a capacity function $c(u)$ is defined. Similarly, a capacity function $c(e)$ is given for each edge $e \in E$. The maximum charge problem is the main focus in this thesis and can be formulated as follows:

$$\begin{array}{lll} \textit{maximize} & \sum_{e \in E} \pi_e & \\ \textit{subject to} & \sum_{e \in E(u)} \pi_e \leq c(u) & \forall u \in V \\ & \pi_e \leq c(e) & \forall e \in E \\ & \pi_e \geq 0 & \forall e \in E \end{array} \tag{MCP}$$

We will revisit the maximum charge problem in the conclusion of this thesis.

Chapter 3

Linear Programming

3.1 Introduction

A *linear program* (*LP*) has a linear objective function which must be either maximized or minimized while obeying a system of linear equalities and inequalities, called *constraints.* A typical linear program has the form:

$$\begin{aligned} &\textit{minimize} && cx \\ &\textit{subject to} && Ax \geq b \\ & && x \geq 0 \end{aligned} \tag{3.1}$$

where $c, x, b \in \mathbb{R}^n$ represent the co-efficient, the variable, and the right-hand-side vectors respectively. Meanwhile, $A \in \mathbb{R}^m \times \mathbb{R}^n$ represents the constraint matrix of the problem (henceforth, we will use a_i to denote a row, and A_j to denote a column in matrix A). Visually, the equations in $Ax \geq b$ specifies a multi-dimensional polyhedron in which all possible feasible solutions to the LP reside; the optimal solution to the LP is located graphically on one of the corners, along one of the edges, or on one of the faces of this polyhedron [Edm65a]. Equation (3.1) is a *minimization LP problem.* A *maximization LP problem*, on the other hand, has the following form:

$$\begin{aligned} &\textit{maximize} && cx \\ &\textit{subject to} && Ax \leq b \\ & && x \geq 0 \end{aligned} \tag{3.2}$$

In this case, $Ax \leq b$ also forms a polyhedron of high dimension in which all feasible solutions to the minimization LP exist. Linear programs found in the form of (3.1) and (3.2) are said to be expressed in *canonical form.* Meanwhile, linear programs in the form

$$\begin{aligned} max(min)imize \quad & cx \\ subject\ to \quad & Ax = b \\ & x \geq 0 \end{aligned} \tag{3.3}$$

are said to be in *standard form.* In fact, canonical and standard forms are equivalent, meaning that an instance of LP in canonical form can be transformed into standard form (and vice versa) using a series of *surplus* and *slack variables* [PS82].

A given linear program can be solved using an iterative method called the *simplex algorithm.* The simplex was first devised by Dantzig in 1947 and uses a series of column manipulations, or *pivots*, to visit each corner of the multi-dimensional polyhedron in an attempt to discover the optimal solution. The main idea behind the algorithm is that a given LP is first represented in standard form; this means that if the original problem is in canonical form, *slack variables* $(x_{n+1}...x_{n+m})$ will be added to (or subtracted from) each row a_i of the constraint matrix. Each slack variable will also appear in the objective function with co-efficients $c_{n+1}, ..., c_{n+m} = 0$. These variables make up the initial *basic vectors* or *basis*: columns A_j corresponding to variables x_j that can be non-zero while still maintaining the equality in the constraints $Ax = b$. Essentially, the simplex repeatedly checks all columns corresponding to "non-basic" variables to see if a more advantageous solution can be obtained by moving it into the basis. While the simplex algorithm is very general and runs in polynomial time in most cases, it has been shown to perform exponentially poorly in certain extreme cases [KM72].

In 1979, Khachiyan [Kha79] proved that linear programming problems can indeed be solved in polynomial time using what's called an *ellipsoid method.* In this paper however, we will be using a method called the *primal dual algorithm* [DLF56] which is an adaptation of the simplex algorithm. We will first introduce the notion of duality which exists for all linear programming formulations.

3.2 Duality

In this section, we introduce the idea of duality which is closely related to many other concepts in linear programming. The duality principle states that a given LP problem can be solved from two different points of view: the primal and the dual. Every LP, denoted the *primal problem*, has an equivalent *dual.* Consider the following minimization LP:

$$\begin{aligned} &\textit{minimize} && cx \\ &\textit{subject to} && Ax \geq b \\ & && x \geq 0 \end{aligned} \tag{3.4}$$

The corresponding dual is written as:

$$\begin{aligned} &\textit{maximize} && \pi b \\ &\textit{subject to} && \pi A \leq c \\ & && \pi \geq 0 \end{aligned} \tag{3.5}$$

The dual representation provides a bound for the primal problem; for instance, the objective function in (3.5) is a lower bound for the objective function in (3.4). In general:

$$cx \geq (\pi A)x = \pi(Ax) \geq \pi b \tag{3.6}$$

This is known as the *weak duality theorem.*

Theorem 3.2.1 (Weak duality thoerem). *Given feasible solutions $x = [x_1, x_2, ..., x_n]$ to the primal in (3.4) and $\pi = [\pi_1, \pi_2, ...\pi_m]$ to the dual in (3.5), the following holds:*

$$\sum_{j=1}^{n} c_i x_j \geq \sum_{i=1}^{m} \pi_i b_i$$

Proof. From equation (3.6), we can obtain the expanded form

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} \pi_i a_{ij} \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) \pi_i \geq \sum_{i=1}^{m} \pi_i b_i$$

which proves the theorem. □

The primal (dual) objective function values are upper (lower) bounds for the dual (primal) and the *strong duality theorem* states that their optimal objective function values are equal.

Theorem 3.2.2 (Strong duality thoerem [PS82]). *Given a primal representation P and a dual representation D of an LP problem, if an optimal solution $x^* = [x_1^*, x_2^*, ..., x_n^*]$ exists for P, then a corresponding optimal $\pi^* = [\pi_1^*, \pi_2^*, ..., \pi_m^*]$ exists for D. Furthermore,*

$$\sum_{j=1}^{n} c_i x_j^* = \sum_{i=1}^{m} \pi_i^* b_i \qquad \square$$

From the strong and weak duality theorems, we see that a delicate balance exists between the primal and dual solutions. Given primal-dual pairs x and π, the conditions needed to be satisfied for both to be at optimal is stated in *the complementary slackness theorem.*

Theorem 3.2.3 (Complementary slackness thoerem). *A feasible primal-dual pair x, π is optimal if and only if*

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} \pi_i a_{ij} - c_j \right) x_j = 0 \tag{3.7}$$

$$\sum_{i=1}^{m} \left(b_i - \sum_{j=1}^{n} a_{ij} x_j \right) \pi_i = 0 \tag{3.8}$$

To be more specific, each term in the summation of the complementary slackness theorem must be equal to 0. Therefore, the necessary and sufficient conditions for x and π can also be written as

$$(\pi A_j - c_j)\, x_j = 0 \qquad j = 1, ..., n$$
$$(b_i - a_i x)\, \pi_i = 0 \qquad i = 1, ..., m$$

This is a crucial theorem in the discussion of the primal-dual algorithm.

3.3 Primal-Dual Algorithm

This section introduces the concepts behind the primal-dual algorithm, which is a general method for solving LP's. The method was first conceived by Kuhn in 1955 [Kuh55] in an attempt to solve the assignment problem. Coincidentally, the assignment problem is nothing more than the weighted bipartite matching problem. The algorithm was later generalized by Ford and Fulkerson in 1958 [FF58].

The primal-dual algorithm, starts off with a feasible solution to the dual π. Then, through gradual improvements to π, it tries to obtain a feasible solution to the primal problem in standard form. The following is a detailed description of the algorithm. Our presentation is based on Papadimitriou and Steiglitz [PS82].

Consider a linear programming primal problem in standard form

$$\begin{aligned} &\textit{minimize} && cx \\ &\textit{subject to} && Ax = b \\ & && x \geq 0 \end{aligned} \tag{P}$$

where $b_i \geq 0$ for all $i = 1, ..., n$. This can be ensured by simply multiplying by -1 all equalities in which $b_i < 0$ in the original form. Next, we will construct the dual

$$\begin{aligned} &\textit{maximize} && \pi b \\ &\textit{subject to} && \pi A \leq c \\ & && \pi \gtrless 0 \end{aligned} \tag{D}$$

The unconstrained variable π in the dual is due to the equality that we started with in the primal. We must now revisit the complementary slackness theorem from the previous section

$$(\pi A_j - c_j)\, x_j = 0 \qquad j = 1, ..., n \tag{3.9}$$

$$(b_i - a_i x)\, \pi_i = 0 \qquad i = 1, ..., m \tag{3.10}$$

Since P is in standard form, any feasible solution x in P will satisfy all conditions in equation (3.10). So, we must now turn our attention to the conditions in (3.9).

The main idea behind the primal-dual algorithm is that we begin with a feasible solution π to the dual and attempt to work towards feasibility in the primal. Given a feasible π, we will define the set J of "tight relations" in the dual constraints as follows:

$$J = \{j : \pi A_j = c_j\}.$$

Relations of this kind will allow us the flexibility to set the corresponding primal variable x_j to any arbitrary positive value while satisfying equation (3.9). On the other hand, relations not found in J will force us to set the corresponding primal variables to $x_j = 0$. Next, in an attempt to satisfy the constraints in (D) which are not in J, we must revisit the primal in an attempt to gain feasibility. This is done by constructing a new LP called the *restricted primal* or *RP*:

$$\begin{array}{lll} \textit{minimize} & \displaystyle\sum_{i=1}^{m} s_i & \\ \textit{subject to} & \displaystyle\sum_{j\in J} a_{ij}x_j + s_i = b_i & i = 1, ..., m \\ & x_j \geq 0 & j \in J \\ & x_j = 0 & j \notin J \\ & s_i \geq 0 & i = 1, ..., m \end{array} \tag{RP}$$

In other words, for each constraint in (P), a slack variable s_i is added in an attempt to gain feasibility in the primal. Notice that when $s_i = 0$ for all $i = 1, ..., n$, then we are indeed at a feasible and optimal solution in P. However, if we are unable to find such a solution we consider the *dual of the restricted primal* or *DRP*:

$$\begin{array}{lll} \textit{maximize} & \pi b & \\ \textit{subject to} & \pi A_j \leq 0 & j \in J \\ & \pi_i \leq 1 & i = 1, ..., m \\ & \pi_i \gtrless 0 & \end{array} \tag{DRP}$$

The DRP is a simplified version of the dual and can sometimes be solved by simpler combinatorial algorithm. Henceforth, we will denote a feasible solution to the DRP as $\overline{\pi}$ and an optimal solution to the DRP as $\overline{\pi}_{opt}$. The dual of the restricted primal suggests that an augmentation of the present dual solution π is possible by a linear combination of π and $\overline{\pi}_{opt}$:

$$\pi^* = \pi + \theta\overline{\pi}_{opt} \tag{3.11}$$

where π^* represents the new dual solution. We will first notice that given an optimal solution $\overline{\pi}_{opt}$ to DRP, it is always the case that $\overline{\pi}_{opt}b > 0$ since this coincides with the optimal solution to the RP. Therefore, $\pi^*b > \pi b$ for any value of $\theta > 0$. The question now becomes what is the upper bound on θ?

To examine this question, we must figure out how much we can increase θ such that π^* is still feasible by discovering the column which becomes "tight" in the dual constraints. In doing this, A_j is multiplied to both sides of equation (3.11).

$$\pi^* A_j = \pi A_j + \theta \overline{\pi}_{opt} A_j \tag{3.12}$$

Now, consider the DRP constraints for the set J:

$$\overline{\pi} A_j \leq 0 \qquad \forall j \in J \tag{3.13}$$

It is obvious that these columns do not limit θ since

$$\pi A_j + \theta \overline{\pi}_{opt} A_j \leq c_j \tag{3.14}$$

for any $\theta > 0$. Our attention must now focus on the columns not in the set J. Once again, any column $j \notin J$ such that $\overline{\pi} A_j \leq 0$ will not figure in the limitation of θ. In fact, if $\overline{\pi} A_j \leq 0$ for all $j \notin J$ then the dual problem D is unbounded because there will be no limit on θ.

Finally, we conclude that if there is some $j \notin J$ such that $\overline{\pi} A_j > 0$ then we can solve for θ according to equation (3.14):

$$\theta = \min_{j \notin J,\, \overline{\pi}_{opt} A_j > 0} \left\{ \frac{c_j - \pi A_j}{\overline{\pi}_{opt} A_j} \right\} \tag{3.15}$$

Notice that the equation for θ ensures that at least one column corresponding to the primal variable x_i enters the set of tight relations J at the beginning of the next iteration. We can now improve our dual solution as in equation (3.11). This process is repeated until an RP-DRP pair whose optimal objective values are 0 is found.

3.3.1 Primal-Dual Example

In this section, we will observe the primal-dual algorithm in action by applying it to the shortest path problem; this is based on an example given in Papadimitriou and Steiglitz [PS82]. Given a weighted directed graph $G(V, E)$, the *shortest path* between two vertices s and t in V is a path from s to t in G such that the sum of the edge weights is minimized.

The shortest path problem itself will be formulated as the primal problem:

$$\begin{array}{ll} \textit{minimize} & cx \\ \textit{subject to} & Ax = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}_m \\ & x \geq 0 \end{array} \tag{P$_{\text{sp}}$}$$

where A is the $m \times n$ incidence matrix of G (assuming G has n edges and m vertices). The value $A_{ij} = +1$ ($A_{ij} = -1$) if a directed edge e_i leads into (away from) vertex u_j; otherwise $A_{ij} = 0$. The vector $x \in \mathbb{R}^n$ represents the edge variables while the vector $c \in \mathbb{R}^n$ represents the weights for each edge. The $+1$ and -1 values in the constraints represent the vertices s and t respectively. The dual of the shortest path problem is written as follows:

$$\begin{array}{lll} \textit{maximize} & \pi_s - \pi_t & \\ \textit{subject to} & \pi_u - \pi_v \leq c_e & \forall e = (u,v) \in E \\ & \pi_u \gtrless 0 & \forall u \in V \end{array} \tag{D$_{\text{sp}}$}$$

Note that the objective function is simplified to $\pi_s - \pi_t$ since s and t are the only non-zero coefficients in (P$_{\text{sp}}$). Given a feasible solution π to (D$_{\text{sp}}$), we will define the set J of tight relations in D as:

$$J = \{e = (u,v) : \pi_u - \pi_v = c_e\}.$$

From the set J we can now construct the restricted primal as

$$\begin{array}{lll} \textit{minimize} & \sum_{u \in V} s_u & \\ \textit{subject to} & A^J x^J + \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_{m-1} \\ s_m \end{bmatrix}_m = \begin{bmatrix} +1 \\ 0 \\ \vdots \\ 0 \\ -1 \end{bmatrix}_m & \\ & x_e \geq 0 & \forall e \in J \\ & s_u \geq 0 & \forall u \in V \end{array} \tag{RP$_{\text{sp}}$}$$

where A^J is a $m \times |J|$ matrix corresponding to the columns in J from A and x^J is the variable vector x containing only variables corresponding to J. Finally, we can look at the DRP for this problem:

$$\begin{array}{llll} \textit{maximize} & \pi_s - \pi_t & & (\text{DRP}_{\text{sp}}) \\ \textit{subject to} & \pi_u - \pi_v \leq 0 & \forall e = (u,v) \in J & \\ & \pi_u \leq 1 & \forall u \in V & \\ & \pi_u \gtrless 0 & \forall u \in V & \end{array}$$

The definition of (DRP_{sp}) states that all "tight edges" in the set J will appear in the first constraint. This suggests that in order to maximize this objective function vlaue, we should give a DRP value $\pi_s = 1$ and all vertices that are reachable from s through edges in J:

$$(\overline{\pi}_{opt})_u = \left\{ \begin{array}{cc} 1 & u = s \text{ or is reachable from } s \text{ through edges in J} \\ 0 & \text{otherwise} \end{array} \right\} \tag{3.16}$$

The final step of the primal-dual algorithm is to find θ given an optimal solution $\overline{\pi}_{opt}$ to (DRP_{sp}). For the shortest path problem, θ is expressed as:

$$\theta = \min_{e=(u,v)\notin J,\ (\overline{\pi}_{opt})_u - (\overline{\pi}_{opt})_v > 0} \{c_e - (\pi_u - \pi_v)\} \tag{3.17}$$

Notice that the denominator from equation (3.15) will always be 1 since any $e = (u,v) \notin J$ such that $\overline{\pi}_u - \overline{\pi}_v > 0$ will have the property that $\overline{\pi}_u - \overline{\pi}_v = 1$ according to equation (3.16). The new feasible dual solution π^* will now be

$$\pi^* = \pi + \theta\overline{\pi}$$

The process of improving the dual feasible solution π and finding the corresponding π_{opt} in the DRP will be repeated until we come across an instance of DRP_{sp} that has the optimal objective function

$$(\overline{\pi}_{opt})_s - (\overline{\pi}_{opt})_t = 0$$

At this point, we are at an optimal and feasible solution to P. For a detailed example of this problem, please see appendix A.2

Chapter 4

Unconstrained Fractional Matching

In this chapter, we will investigate a modification of the maximum charge problem using the primal dual algorithm. This new problem, which we have named *Maximum Unconstrained Fractional Matching*, does not put any constraint on edge values π_e for all $e \in E$; in other words, the final two constraints in (MCP) are replaced by $\pi_e = [-\infty, +\infty]$. Appendix A.3 shows that allowing negative edge values, at times, can give larger objective function values.

4.1 Problem Formulation

As stated in the primal dual algorithm (section 3.3), we will first impose equality on the first set of constraints in the primal problem. Given a graph $G(V, E)$, the primal is defined as

$$\begin{array}{lll} \textit{minimize} & \sum\limits_{u \in V} c_u x_u & \\ \textit{subject to} & x_u + x_v = 1 & \forall (u,v) \in E \\ & x_u \geq 0 & \forall u \in V \end{array} \qquad (\mathrm{P_{ufm}})$$

For each vertex u, c_u is the weight of u and x_u is the value assigned to u. It is worth noting that in this case, the matrix A in the general definition of the primal-dual formulation is once again the vertex-edge incidence matrix of the graph G. The dual problem is the unconstrained fractional matching problem:

$$\begin{array}{lll} \textit{maximize} & \sum_{e\in E} \pi_e & \\ \textit{subject to} & \sum_{e\in E(u)} \pi_e \leq c_u & \forall u \in V \\ & \pi_e \gtrless 0 & \forall e \in E \end{array} \qquad (\mathrm{D}_{\mathrm{ufm}})$$

where π_e is the value assigned to edge e and $E(u)$ is the set of adjacent edges of vertex u (these definitions will be formalized in the following section). Given a feasible solution π to the dual representation $(\mathrm{D}_{\mathrm{ufm}})$, the set J can be defined as

$$J = \left\{ u : \sum_{e\in E(u)} \pi_e = c_u \right\}$$

From this, we will define our restricted primal using a slack variable s_e for each edge:

$$\begin{array}{lll} \textit{minimize} & \sum_{e\in E} s_e & \\ \textit{subject to} & x_u + x_v + s_e = 1 & \forall (u,v) \in E \\ & s_e \geq 0 & \forall e \in E \\ & x_u \geq 0 & \forall u \in J \\ & x_u = 0 & \forall u \notin J \end{array} \qquad (\mathrm{RP}_{\mathrm{ufm}})$$

From the restricted primal, we derive the dual of the restricted primal

$$\begin{array}{lll} \textit{maximize} & \sum_{e\in E} \pi_e & \\ \textit{subject to} & \sum_{e\in E(u)} \pi_e \leq 0 & \forall u \in J \\ & \pi_e \gtrless 0 & \forall e \in E \\ & \pi_e \leq 1 & \forall e \in E \end{array} \qquad (\mathrm{DRP}_{\mathrm{ufm}})$$

4.2 Definitions

We will use the following notation for our definitions:

- $\overline{\pi}$ denotes a feasible solution to a current instance of ($\mathrm{DRP}_{\mathrm{ufm}}$)
- π denotes a feasible solution to the current instance of the dual problem ($\mathrm{D}_{\mathrm{ufm}}$)
- $E(u)$ denotes the set of all edges adjacent to the vertex u
- Given a feasible DRP solution $\overline{\pi}$ and a vertex u, the *adjacency function* $\overline{\pi}(u)$ is defined as $\sum_{e \in E(u)} \overline{\pi}_e$

Saturated and unsaturated vertices:
Given a feasible solution π to the dual problem, a vertex u is *unsaturated with respect to* π if

$$\sum_{e \in E(u)} \pi_e < c_u.$$

On the other hand, a *saturated vertex* u *with respect to* π has the property that

$$\sum_{e \in E(u)} \pi_e = c_u.$$

Saturated and unsaturated edges:
Given a feasible solution $\overline{\pi}$ to the DRP, an edge e *is unsaturated with respect to* $\overline{\pi}$ if $\overline{\pi}_e < 1$. If $\overline{\pi}_e = 1$, edge e is *saturated with respect to* $\overline{\pi}$.

Alternating walk (path and lasso):
An *alternating walk* P $(u_1, e_1, u_2, e_2, ..., u_t, e_t, u_{t+1})$ of length t *with respect to a feasible solution* $\overline{\pi}$ is a sequence of vertices and edges such that each vertex u_i is connected to u_{i+1} through edge e_i in the graph G (for $i = 0...t$). An alternating walk P has the following properties:

1. P begins and ends at unsaturated vertices u_1 and u_{t+1} with respect to π.
2. All **intermediate vertices** $u_2, ..., u_t$ are saturated with respect to π.
3. All odd edges in P $(e_1, e_3, e_5...)$ are unsaturated with respect to $\overline{\pi}$. In other words, $\overline{\pi}_{e_{2i-1}} < 1$ $(i = 0...\frac{t+1}{2})$.

An alternating walk P is an *alternating path* if vertices in P are pairwise distinct; otherwise, it is an *alternating lasso.*

Augmenting path and lasso:
An alternating walk $P = (u_1, e_1, u_2, e_2, ..., u_t, e_t, u_{t+1})$ is *augmenting* if it is of odd length.

Furthermore, if $\{u_1, u_2, ..., u_{t+1}\}$ are distinct, then P is an *augmenting path*; otherwise it is an *augmenting lasso.*

Given a feasible $\overline{\pi}$, we will define *"augmenting a path or lasso P by ϵ"* as the process of adding ϵ to all the odd edges $\overline{\pi}_{e_{2i-1}}$ of P while subtracting ϵ from all the even edges $\overline{\pi}_{e_{2i}}$ of P. In an augmenting path, odd edges are referred to as **plus edges** and even edges as **minus edges**. Given an augmenting path P, augmenting P by ϵ will increase the overall DRP solution $\overline{\pi}$ by ϵ. Also note that augmenting a path or lasso P does not change the value of the adjacency function for each intermediate vertex u in P (this will be formalized in lemma 4.3.2).

4.3 Main Theorem

In this section, we will prove that we can derive an optimal solution $\overline{\pi}$ to an instance of (DRP_{ufm}) by searching for augmenting paths/lassos and augmenting them until all such paths/lassos have been exhausted. We will begin by showing that a feasible solution $\overline{\pi}$ to (DRP_{ufm}) is not optimal if there exists an augmenting path/lasso with respect to $\overline{\pi}$.

Lemma 4.3.1. *Given a feasible solution $\overline{\pi}$ to an instance of* (DRP_{ufm}), *if there exists an augmenting path/lasso P $(u_1, e_1, u_2, e_2, ..., e_{2k-1}, u_{2k})$ with respect to $\overline{\pi}$, then $\overline{\pi}$ is not optimal.*

Proof. Assume that in traversing P, edge e_i is visited no more than t_{e_i} times. Furthermore, let:

$$d = \min\left\{\frac{1-\overline{\pi}_{e_1}}{t_{e_1}}, \frac{1-\overline{\pi}_{e_3}}{t_{e_3}}, ..., \frac{1-\overline{\pi}_{e_{2k-1}}}{t_{e_{2k-1}}}\right\} \tag{4.1}$$

that is, d is the odd edge e such that $\frac{1-\overline{\pi}_e}{t_e}$ is minimized. Notice that d must be positive since all $\overline{\pi}_e$ must be strictly less than 1 (by the definition of augmenting path/lasso). Now, augment the path/lasso P by adding $d * t_{e_{2i-1}}$ to all odd edges e_{2i-1} and subtracting $d * t_{e_{2i}}$ from all even edges e_{2i}. Clearly, the new $\overline{\pi}$ is still feasible since d ensures that no $\overline{\pi}_e$ goes above 1. After the augmentation, we have increased $\overline{\pi}$ by the amount $d{*}t_{e_1}$. □

We should first note that augmenting a path/lasso P by the amount d does not change $\sum_{e \in E(u)} \overline{\pi}_e$, the sum of the edge values adjacent on any vertex $u \in J$. This will ensure that the cardinality of set J does not decrease after each iteration of the primal-dual algorithm (see section 4.3.1). In other words, the saturated vertices at the end of one iteration will also be saturated at the beginning of the next iteration. This is formalized in the following lemma.

Lemma 4.3.2. *Given a feasible solution $\overline{\pi}$ to* (DRP$_{\text{ufm}}$) *and an augmenting path/lasso $P = (u_1, e_1, u_2, e_2, ..., u_t, e_t, u_{t+1})$, assume that the augmentation process described in lemma 4.3.1 was applied to P, resulting in a new feasible solution $\overline{\pi}'$. Then*

$$\overline{\pi}'(u) - \overline{\pi}(u) = 0$$

for all of the intermediate vertices $u = u_2, ..., u_t$.

Proof. Given an intermediate vertex u, the augmenting amount d found by lemma 4.3.1 is added to and subtracted from edges on either side of u. Hence the net total $\overline{\pi}'(u) - \overline{\pi}(u)$ is 0. □

Before continuing further, we will show that the absence of augmenting paths/lassos with respect to $\overline{\pi}$ is a necessary and sufficient condition for optimality.

Theorem 4.3.3. *(structural theorem) Let $\overline{\pi}$ be a feasible solution to an instance of* (DRP$_{\text{ufm}}$) *found by repeatedly finding augmenting paths/lassos and augmenting them starting at an initial solution of $\overline{\pi} = 0$. Then, $\overline{\pi}$ is not optimal if and only if there exists an augmenting path/lasso with respect to $\overline{\pi}$ in the graph G.*

Proof. We will begin by noting that the converse statement has already been proved by lemma 4.3.1: if there exists an augmenting path with respect to $\overline{\pi}$ then $\overline{\pi}$ is not optimal. Thus, it is left to show that *if the solution $\overline{\pi}$ is not optimal, then there exists an augmenting path with respect to $\overline{\pi}$.*

To prove this, we will use a technique similar to [KGGS06][1]. First, make the following assumptions:

(i) $\overline{\pi}$ is a feasible solution to the DRP instance which is *not optimal.*

(ii) $\overline{\pi}^*$ is an optimal solution to the DRP instance.

[1]The proof's framework is attributed to Horst Sachs, and was used to prove the structure theorem of the maximum charge problem.

(iii) Among all maximum solutions, $\overline{\pi}^*$ minimizes $\sum_{e \in \mathrm{E}} |\overline{\pi}^*_e - \overline{\pi}_e|$

Now, we can use $\overline{\pi}^*$ to find an augmenting path with respect to $\overline{\pi}$. Since $\overline{\pi}^*$ satisfies assumption (ii), it must be true that

$$\sum_{e \in \mathrm{E}} \overline{\pi}_e = \frac{1}{2} \sum_{u \in \mathrm{V}} \sum_{e \in \mathrm{E(u)}} \overline{\pi}_e < \frac{1}{2} \sum_{u \in \mathrm{V}} \sum_{e \in \mathrm{E(u)}} \overline{\pi}^*_e = \sum_{e \in \mathrm{E}} \overline{\pi}^*_e$$

Thus, there must exist a vertex u_1 such that

$$\sum_{e \in \mathrm{E}(u_1)} \overline{\pi}_e < \sum_{e \in \mathrm{E}(u_1)} \overline{\pi}^*_e$$

Furthermore, this implies that there must be an edge e_1 from u_1 to u_2 such that

$$\overline{\pi}_{e_1} < \overline{\pi}^*_{e_1}$$

Now, we can recursively define a general process to find an augmenting path depending on whether we are presently at an odd or even edge.

1. *Odd edge step.* Currently, we have found an odd edge e_1 from u_1 to u_2 (in general, an odd edge e_{2i-1} from u_{2i-1} to u_{2i}); if the new adjacent vertex has already been visited (i.e. $u_{2i} \in \{u_1, u_2, ..., u_{2i-1}\}$), then **stop**. Otherwise, if u_2 (u_{2i}) also has the property that

$$\sum_{e \in \mathrm{E}(u_2)} \overline{\pi}_e < \sum_{e \in \mathrm{E}(u_2)} \overline{\pi}^*_e \qquad \Big(\sum_{e \in \mathrm{E}(u_{2i})} \overline{\pi}_e < \sum_{e \in \mathrm{E}(u_{2i})} \overline{\pi}^*_e \Big)$$

then u_2 (u_{2i}) is unsaturated with respect to π because if u_2 ($u_2 i$) was a saturated vertex (intermediate vertex in an augmentation), there would not be an inequality (see lemma 4.3.2); we can **stop**. Otherwise, u_2 (u_{2i}) has the property that

$$\sum_{e \in \mathrm{E}(u_2)} \overline{\pi}_e \geq \sum_{e \in \mathrm{E}(u_2)} \overline{\pi}^*_e \qquad \Big(\sum_{e \in \mathrm{E}(u_{2i})} \overline{\pi}_e \geq \sum_{e \in \mathrm{E}(u_{2i})} \overline{\pi}^*_e \Big).$$

This implies that there must be an edge e_2 from u_2 to u_3 (in general an edge e_{2i} from u_{2i} to u_{2i+1}) such that

$$\overline{\pi}_{e_2} > \overline{\pi}^*_{e_2} \qquad (\overline{\pi}_{e_{2i}} > \overline{\pi}^*_{e_{2i}}).$$

2. *Even edge step.* We are now at an even edge e_2 from u_2 to u_3 (in general, e_{2i} from u_{2i} to u_{2i+1}); if the new adjacent vertex has already been visited (i.e. $u_{2i+1} \in \{u_1, u_2, ..., u_{2i}\}$), then **stop**. Furthermore, if u_3 (u_{2i+1}) has the property that

$$\sum_{e \in \mathrm{E}(u_3)} \overline{\pi}_e > \sum_{e \in \mathrm{E}(u_3)} \overline{\pi}^*_e \qquad \Big(\sum_{e \in \mathrm{E}(u_{2i+1})} \overline{\pi}_e > \sum_{e \in \mathrm{E}(u_{2i+1})} \overline{\pi}^*_e \Big)$$

then u_3 (u_{2i+1}) is unsaturated with respect to π; we can **stop**. Otherwise, u_3 (u_{2i+1}) will have the property that

$$\sum_{e \in \mathrm{E}(u_3)} \overline{\pi}_e \leq \sum_{e \in \mathrm{E}(u_3)} \overline{\pi}^*_e \qquad \Big(\sum_{e \in \mathrm{E}(u_{2i+1})} \overline{\pi}_e \leq \sum_{e \in \mathrm{E}(u_{2i+1})} \overline{\pi}^*_e \Big).$$

Once again this suggests that there is an odd edge e_3 (in general e_{2i+1}) such that

$$\overline{\pi}_{e_3} < \overline{\pi}^*_{e_3} \qquad (\overline{\pi}_{e_{2i+1}} < \overline{\pi}^*_{e_{2i+1}})$$

We are now back at an odd edge, iterate back to step 1 above until the procedure terminates.

When this procedure stops, we have found a path or lasso P from one unsaturated vertex (with respect to π) to another. There are two cases for P:
Case 1: P is a path or lasso of even length. If this is the case, we can decrease (increase) $\overline{\pi}^*_e$ for odd (even) edges e, thereby decreasing $\sum_{e \in \mathrm{E}} |\overline{\pi}^*_e - \overline{\pi}_e|$. This contradicts assumption (iii) and can never happen.
Case 2: P is a path or lasso of odd length. If this is the case, we have found an augmenting path or lasso. □

4.3.1 Naïve Algorithm Analysis

Given an instance of ($\mathrm{DRP_{ufm}}$) and its corresponding set of saturated vertices J (with respect to π, theorem 4.3 suggests a naïve algorithm to solve ($\mathrm{DRP_{ufm}}$) by repeatedly looking for augmenting paths/lassos and augmenting them. This method is outlined in algorithm 4.1.

Essentially, for a given instance of ($\mathrm{DRP_{ufm}}$), the naïve algorithm finds all the unsaturated vertices with respect to π. In the first stage, it uses a depth first search to find and augment all augmenting paths leading away from each unsaturated vertex u (we call this method **DFS_paths**($s, \overline{\pi}, J$) in algorithm 4.1). In the second stage, the algorithm will deploy another depth first search, this time looking for any augmenting lassos that remain and augmenting them (in algorithm 4.1, we call this method **DFS_lassos**($s, \overline{\pi}, J$)).

Theorem 4.3.4. *All augmenting lassos found in the second stage of the naïve algorithm are plus edge disjoint.*

Proof. We prove this by contradiction. After the first stage of the naïve algorithm, all augmenting paths will be found and augmented, resulting in a feasible solution $\overline{\pi}$ for the DRP. Now, assume that there are two augmenting lassos which share a plus edge. Figure 4.1 shows three situations in which this could happen (the highlighted vertices represent unsaturated vertices with respect to π). In the leftmost figure, since blossoms are odd length cycles, there must exist an odd length augmenting path between a and b through either (u,v) or (u,v'). This contradicts the assumption that all augmenting paths have been found by the first stage of the algorithm. In the center figure, we see that two augmenting lassos share a common plus edge within their respective blossoms. Once again, since blossoms are odd length cycles, there must exist an augmenting path from vertex c to vertex d through either (u,v) or (u,v'); contradicting the original assumption. Finally, in the rightmost figure, we see that two lassos share a plus edge in the path leading to the blossom. In this case, only one of the two lassos (from vertex e or vertex f) can be augmented because one lasso augmentation will saturate the edge (u,v). □

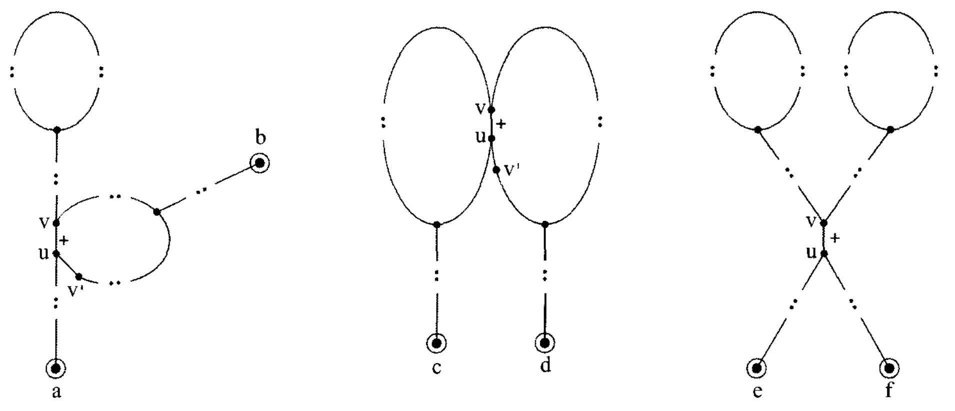


Figure 4.1: three cases in which augmenting lassos can share a plus edge (u,v)

Theorem 4.3.4 suggests that the DRP solution $\overline{\pi}$ will increase by 1 ($\frac{1}{2}$) with each augmenting path (lasso) found by the depth first search and since there must be a finite solution to (DRP_{ufm}), we conclude that this naïve algorithm will terminate in finite time.

Algorithm 4.1: Naïve DRP algorithm

Define:
J is the set of saturated vertices w.r.t π for the current instance of (DRP_{ufm})
extract(U) randomly extracts and removes a vertex in the set U; returns $\varnothing$ if U is empty.
DFS_paths($s, \overline{\pi}, J$) will find and augment an augmenting path from unsaturated vertex s to another unsaturated vertex $u \notin J$, returning true. If one cannot be found, then it will simply return false.
DFS_lassos($s, \overline{\pi}, J$) will find and augment an augmenting lasso from unsaturated vertex s, returning true. If one cannot be found, then it will simply return false.

DRPopt(J):

```
$\overline{\pi} = 0$
$U_1 = V[G] - J$
$U_2 = U_1$
while $U_1 \neq \varnothing$ do
    $s = extract(U_1)$
    repeat
        $found\_path$ = DFS_paths($s, \overline{\pi}, J$)
    until $found\_path = false$ ;
while $U_2 \neq \varnothing$ do
    $s = extract(U_2)$
    repeat
        $found\_lasso$ = DFS_lassos($s, \overline{\pi}, J$)
    until $found\_lasso = false$ ;
```

Theorem 4.3.5. *Given an instance of* (DRP_{ufm}), *algorithm 4.1 finds the optimal solution* $\overline{\pi}_{opt}$ *in* $O(|E|^2)$ *time.*

Proof. We will first note that since DRP edge values $\overline{\pi}_e$ cannot exceed 1, the maximum value of the objective function to (DRP_{ufm}) is at most $|E|$. Since the objective function increases by 1 ($\frac{1}{2}$) for each augmenting path (lasso), we see that the number of augmenting paths/lassos will be $O(|E|)$.

Finally, **DFS_paths**($s, \overline{\pi}, J$) (**DFS_lassos**($s, \overline{\pi}, J$)), which searches for an augmenting path (lasso), will take at most $O(|E|)$ and the theorem follows. □

After the optimal solution $\overline{\pi}$ to the DRP is calculated, we must find a multiplying factor θ

with which to improve the dual solution π. In this case, we have

$$\theta = \min_{u \notin J,\ \overline{\pi}(u)>0} \left\{ \frac{c_u - \sum_{e\in E(u)} \pi_e}{\sum_{e\in E(u)} \overline{\pi}_e} \right\} \tag{4.2}$$

According to equation 4.2, θ is determined by the unsaturated vertex with respect to π that is closest to being saturated. We see that the vertex u that gives θ its value will be in the set J at the beginning of the next iteration.

We are now ready to define the algorithm for finding the optimal solution to an instance of the unconstrained fractional matching problem: we begin with an initial dual solution $\pi = 0$. Then, using algorithm 4.1, we obtain $\overline{\pi}_{opt}$, an optimal solution to the corresponding DRP instance; this $\overline{\pi}_{opt}$ is used to improve the dual solution π and a new instance of (DRP_{ufm}) is obtained. This process is repeated until the objective function in the DRP instance equals 0, meaning an optimal solution π has been obtained. The detailed method is outlined in algorithm 4.2.

Algorithm 4.2: primal-dual interpretation of UFM

Given: an undirected graph $G(V, E)$,

Define:
constructDRP(π) finds the set J given a feasible dual solution π.
calculateθ($\overline{\pi}, \pi, J$) calculates θ according to equation (4.2)
DRPopt(J) runs algorithm 4.1, obtains $\overline{\pi}_{opt}$ to the current DRP instance J.

UFM(G):

```
1  $\pi = 0$
2  $\overline{\pi} = 0$
3  repeat
4      $J = constructDRP(\pi)$
5      $\overline{\pi} = \mathbf{DRPopt}(J)$
6      if $\sum_{e\in E} \overline{\pi}_e = 0$ then
7          return $\pi$
8      else
9          $\theta = calculate\theta(\overline{\pi}, \pi, J)$
10         $\pi = \pi + \theta\overline{\pi}$
11 until $\sum_{e\in E} \overline{\pi}_e = 0$ ;
```

We will first bound the number of iterations of algorithm 4.2. By lemma 4.3.2 we see that during a given iteration of the algorithm, an optimal solution $\overline{\pi}_{opt}$ to the current instance of ($\mathrm{DRP}_{\mathrm{ufm}}$) will be found. According to lemma 4.3.2, the summation of the values assigned to edges incident on any intermediate vertex u ($u \in J$) will be 0. In other words, vertices entering the set J in step 4 of the algorithm will never come out of J until an optimal solution π is found. Furthermore, the calculation of θ in equation (4.2) guarantees that at least one unsaturated vertex (with respect to π) will be saturated at the beginning of the next iteration. Thus this algorithm will repeat $O(|V|)$ times.

This, combined with the results of theorem 4.3.1 allows us to conclude that the overall running time of the naïve algorithm for the unconstrained fractional matching is $O(|V||E|^2)$.

4.4 The Phased Algorithm

We can do better than the naïve algorithm described in the previous section. In this section, we present a "phased" technique for solving an instance of ($\mathrm{DRP}_{\mathrm{ufm}}$) similar to the Hopcroft and Karp algorithm for bipartite matching [HK73]. We will begin by reducing a general graph to a bipartite graph.

4.4.1 Bipartite Graph Reduction

Given a general graph $G(V, E)$, we construct a bipartite graph $G^B(V_1, V_2, E^B)$. We will define $V_1 = \{v_1 \ : \ v \in V\}$ and $V_2 = \{v_2 \ : \ v \in V\}$. Furthermore, for every edge $(u, v) \in E$, there are two edges (u_1, v_2) and (u_2, v_1) in E^B. Next, we define $c(u_1) = c(u_2) = c(u)$ for all $u_1(u_2) \in V_1(V_2)$. Given a general graph G, and its corresponding bipartite reduction G^B, the set of feasible assignments of edge values π^B to G^B has a one-to-one mapping onto the set of feasible assignments of edge values π to G. Given a feasible solution π^B in G^B, for edges $(u_1, v_2), (u_2, v_1) \in E^B$, the corresponding edge $(u, v) \in G$ is assigned the value:

$$\pi_{(u,v)} = \frac{1}{2}\pi^B_{(u_1,v_2)} + \frac{1}{2}\pi^B_{(v_1,u_2)} \tag{4.3}$$

Furthermore, since π^B is a feasible solution, for each pair of corresponding vertices u_1 and u_2 in G^B, we have

$$\sum_{e \in E(u1)} \pi^B_e \leq c(u_1) \text{ and } \sum_{e \in E(u2)} \pi^B_e \leq c(u_2) \tag{4.4}$$

Now, adding the two equations in (4.4), we obtain

$$\sum_{e \in E(u1)} \pi^B_e + \sum_{e \in E(u2)} \pi^B_e \leq c(u_1) + c(u_2) \tag{4.5}$$

Since $c(u_1) = c(u_2) = c(u)$, equation (4.5) can be expanded to

$$\pi^B_{(u_1,(v_1)_2)} + \pi^B_{(u_1,(v_2)_2)} + \ldots + \pi^B_{(u_1,(v_k)_2)} + \pi^B_{(u_2,(v_1)_1)} + \pi^B_{(u_2,(v_2)_1)} + \ldots + \pi^B_{(u_2,(v_k)_1)} \leq 2c(u)$$

$$= \frac{1}{2}(\ \pi^B_{(u_1,(v_1)_2)} + \pi^B_{(u_1,(v_2)_2)} + \ldots + \pi^B_{(u_1,(v_k)_2)} + \pi^B_{(u_2,(v_1)_1)} + \pi^B_{(u_2,(v_2)_1)} + \ldots + \pi^B_{(u_2,(v_k)_1)}\ ) \leq c(u)$$

According to equation (4.3), the edges $(u_1, (v_i)_2)$ and $(u_2, (v_i)_1)$ in G^B contribute $\frac{1}{2}$ to the edge (u, v_i) in G. Thus, we have

$$\sum_{e \in E(u)} \pi_e \leq c(u)$$

This shows that the conversion from π^B to π maintains feasibility.

Similarly, given a feasible solution π in G, for each edge $(u,v) \in E$, the corresponding edges $(u_1, v_2), (u_2, v_1) \in E^B$ are assigned values:

$$\pi^B_{(u_1,v_2)} = \pi^B_{(u_2,v_1)} = \pi_{(u,v)} \tag{4.6}$$

Furthermore, since π is a feasible solution and $c(u_1) = c(u_2) = c(u)$, we have that

$$\sum_{e \in E(u1)} \pi^B_e \leq c(u_1) \text{ and } \sum_{e \in E(u2)} \pi^B_e \leq c(u_2)$$

Once again, we see that the conversion from π to π^B maintains feasibility.

Lemma 4.4.1. *Given an optimal solution π to* ($\mathrm{D_{ufm}}$) *in graph G, with objective function*

$$\sum_{e \in E} \pi_e = Q$$

there must exist an optimal solution π^B to ($\mathrm{D_{ufm}}$) *in the corresponding bipartite graph G^B with objective function*

$$\sum_{e \in E^B} \pi^B_e = 2Q$$

Proof. Assume for the sake of contradiction, that an optimal solution π^{*B} for ($\mathrm{D_{ufm}}$) exists for G^B such that

$$\sum_{e \in E^B} \pi^{*B}_e > \sum_{e \in E^B} \pi^B_e.$$

Then by equation 4.3, there must a solution π for graph G such that

$$\sum_{e \in E} \pi^*_e > \sum_{e \in E} \pi_e.$$

The lemma follows by contradiction. □

The reduction of a general graph G into a bipartite graph G^B does not change the complexity of the running times since it has only increased the number of edges and vertices by a constant factor of 2.

4.4.2 Algorithm

Given a bipartite graph $G^B(V_1, V_2, E^B)$ and a set of saturated vertices J^B, the phased algorithm to solve (DRP$_{\text{ufm}}$) works as follows: we divide our search for augmenting paths into phases. In each phase, we will search for the maximal set of plus edge disjoint augmenting paths of the shortest length from unsaturated vertices in V_1 to unsaturated vertices in V_2 and augment them simultaneously. If no augmenting paths can be found, then by theorem 4.3.3, we are at an optimal solution to (DRP$_{\text{ufm}}$).

To do this, we deploy a breadth first search with all unsaturated vertices in V_1 at level 0. From vertices at an even level $2i$, we obtain saturated vertices (with respect to π) at level $2i + 1$ by following only edges e with $\overline{\pi}_e < 1$ that have not yet been visited (plus edges). From an odd level $2i + 1$, we obtain saturated vertices (with respect to π) in $2i + 2$ through all adjacent edges that have not yet been visited. This process continues until we find an unsaturated vertex in V_2 at an odd level $2t+1$. At this point, a "found_flag" signifies that an augmenting path has been found. The search continues until all unsaturated vertices in V_2 at level $2t + 1$ have been visited. If an unsaturated vertex cannot be found on any odd level then we are at an optimal solution to the DRP. This process is outlined in **BFS**($\overline{\pi}$,J^B,U_1,U_2) of algorithm 4.3.

At this point, we perform an augmentation and topological plus edge delete. We begin at an unsaturated vertex at the last level $2t + 1$ and trace back through the breadth first search tree to obtain an augmenting path P. Path P is augmented in the current solution $\overline{\pi}$ by the amount described in equation (4.1). Each plus edge $e = (u, v)$ in P is deleted by removing u from the parents of v while putting v into a deletion queue Q. The topological delete process then systematically decrements the indegree of all vertices affected by the deletion of edge e; when the indegree of vertex $u \in U_2$ becomes 0, u is deleted from the list of vertices found by the BFS. This process is repeated until there are no more paths to trace back from the last level $2t + 1$. The augmentation and topological plus edge delete, which we call **BFS_visit**($\overline{\pi}$, J^B, U_1, U_2) is outlined in algorithm 4.4.

Algorithm 4.3: BFS search for alternating paths

Given: a bipartite undirected graph $G^B(V_1, V_2, E^B)$,

Define:
J^B is the set of saturated vertices w.r.t π for the graph G^B
$U_1 \subseteq V_1, U_2 \subseteq V_2$: set of unsaturated vertices with respect to π from V_1 and V_2 respectively
$parent[u]$ is a private attribute of vertex u representing a list of parents of vertex u in the BFS
$children[u]$ is an attribute of vertex u representing a list of children of vertex u in the BFS
$level[u]$ is a private attribute of vertex u representing the level of u in the BFS tree.
$status[u] = \{unvisited, finalized\}$ represents the status of vertex u in the BFS process
$found\_vertices\_list$ is a list of vertices in U_2 that have been found by the BFS.

BFS($\overline{\pi}$,J^B,U_1,U_2):

```
for $u \in V_1 \cup V_2$ do
    $status[u] = unvisited$
    $level[u] = \infty$
    $parent[u] = $ NULL
$Q = \varnothing$
for $u \in U_1$ do
    $status[u] = visited$
    $level[u] = 0$
    $parent[u] = $ NULL
    $Enqueue(Q, u)$
$current\_level = 0$
$found\_flag = false$
$u = Dequeue(Q)$
repeat
    while $current\_level = level[u]$ and $Q \neq \varnothing$ do
        for $v \in E(u)$ do
            if $status[v] \neq finalized$ then
                if $is\_even(level[u])$ and $\overline{\pi}_{(u,v)} < 1$ then
                    $level[v] = level[u] + 1$
                    $parent[v] = parent[v] + \{u\}$
                    $children[u] = children[u] + \{v\}$
                    if $v \in U_2$ then
                        $found\_flag = true$
                        $found\_vertices\_list = found\_vertices\_list \cup \{v\}$
                    else
                        $Enqueue(Q, v)$
                else if $is\_odd(level[u])$ then
                    $level[v] = level[u] + 1$
                    $parent[v] = parent[v] + \{u\}$
                    $children[u] = children[u] + \{v\}$
                    $Enqueue(Q, v)$
        $status[u] = finalized$
        $u = Dequeue(Q)$
    $current\_level = current\_level + 1$
until $u = $ NULL or $found\_flag = true$ ;
return $found\_flag$
```

Algorithm 4.4: Augmentation and Topological Plus Edge Delete

Given: a bipartite undirected graph $G^B(V_1, V_2, E^B)$,

Define:
J^B is the set of saturated vertices w.r.t π for the graph G^B
augment$(\overline{\pi}, P)$ augments path P in the current solution $\overline{\pi}$ by the amount in equation (4.1)
found_vertices_list is a list of vertices in U_2 that have been found by the current BFS tree.
parent$[u]$ is an attribute of vertex u representing a list of parents of vertex u in the BFS
children$[u]$ is an attribute of vertex u representing a list of children of vertex u in the BFS

```
BFS_visit($\overline{\pi}$, $J^B$, $U_1$, $U_2$)
$Q = \varnothing$
repeat
    $P$ = traceback($U_1, U_2$)
    if $P \neq \varnothing$ then
        $augment(\overline{\pi}, P)$
        for plus edge $e = (u, v) \in P$ do
            $parent[v] = parent[v] - \{u\}$
            $children[u] = children[u] - \{v\}$
            $Enqueue(Q, v)$
    while $Q \neq \varnothing$ do
        $u = Dequeue(Q)$
        if $|parent[u]| = 0$ then
            if $u \in U_2$ then
                $found\_vertices\_list = found\_vertices\_list - \{u\}$
            else
                for $v \in children[u]$ do
                    $parent[v] = parent[v] - \{u\}$
                    $children[u] = children[u] - \{v\}$
                    $Enqueue(Q, v)$
until $P = \varnothing$ ;
return $\overline{\pi}$
```

```
traceback($U_1$, $U_2$)
$P = \varnothing$
while $found\_vertices\_list \neq \varnothing$ do
    $v = found\_vertices\_list[0]$
    repeat
        $P = P + \{v\}$
        $u = (parent[v])[0]$
        $P = P + \{u, (u, v)\}$
        $v = u$
    until $u \in U_1$ ;
return $P$
```

A general graph G is first reduced to a bipartite graph G^B using the reduction in section 4.4.1. The algorithm finds unsaturated vertices $U_1 \subseteq V_1$ and $U_2 \subseteq V_2$ with respect to a solution π. It then uses $\mathbf{BFS}(\overline{\pi}, J^B, U_1, U_2)$ to find the shortest augmenting paths from U_1 to U_2 before augmenting the maximal set of plus edge disjoint augmenting paths. This process is repeated until there are no more augmenting paths found by the breadth first search. The phased algorithm for ($\mathrm{DRP}_{\mathrm{ufm}}$) is presented in algorithm 4.5 and the corresponding primal-dual interpretation is shown in algorithm 4.6.

Algorithm 4.5: Phased DRP Algorithm

Given: a bipartite undirected graph $G^B(V_1, V_2, E^B)$,

Define:
J^B is the set of saturated vertices w.r.t π for the graph G^B
$\mathbf{BFS}(\overline{\pi}, J^B, U_1, U_2)$ is the alternating path BFS method discussed in algorithm 4.3; returns true if an augmenting path is found, false otherwise.
$\mathbf{BFS\_visit}(\overline{\pi}, J^B, U_1, U_2)$ is the augmentation and topological plus edge delete method discussed in algorithm 4.4

DRPopt2(J^B):

```
1 $\overline{\pi} = 0$
2 $U_1 = V_1 - (J^B \cap V_1)$
3 $U_2 = V_2 - (J^B \cap V_2)$
4 repeat
5     $aug\_path = \mathbf{BFS}(\overline{\pi}, J^B, U_1, U_2)$
6     if $aug\_path$ then
7         $\overline{\pi} = \mathbf{BFS\_visit}(\overline{\pi}, J^B, U_1, U_2)$
8 until $aug\_path = false$ ;
9 return $\overline{\pi}$
```

For the purposes of analyzing the running time, we must show that augmenting paths found by each successive phase is strictly increasing, this will allow us to bound the number of phases for this algorithm. Furthermore, we will argue that for a given phase, we can safely delete plus edges without fear of them being used as minus edges for other augmenting paths in the same phase. These ideas are discussed in the following lemmas:

Lemma 4.4.2. *Augmenting paths found in each successive phase of algorithm 4.5 is strictly increasing.*

Proof. Assume during the first R phases of algorithm 4.5, that the minimum length augmenting paths of successive phases were strictly of increasing order. Furthermore, assume

Algorithm 4.6: primal-dual interpretation of UFM for the phased algorithm

Given: an undirected graph $G(V, E)$,

Define:
constructDRP(π^B) finds the set J^B given a feasible dual solution π^B.
calculate$\theta(\overline{\pi}^B, \pi^B, J^B)$ calculates θ according to equation (4.2)
DRPopt2(J^B) runs algorithm 4.5, obtains $\overline{\pi}^B_{opt}$ to the current DRP instance J^B.
makeBipartite(G) returns the bipartite reduction of graph G as described in section 4.4.1.
convert$(\overline{\pi}^B)$ converts the DRP solution for the bipartite graph into a DRP solution for the general graph as described in equation 4.3.

UFM(G):

```
1  $\pi^B = 0$
2  $\overline{\pi}^B = 0$
3  $G^B = makeBipartite(G)$
4  repeat
5      $J^B = constructDRP(\pi^B)$
6      $\overline{\pi}^B = \textbf{DRPopt2}(J^B)$
7      if $\sum_{e \in E} \overline{\pi}^B_e = 0$ then
8          $\overline{\pi} = convert(\overline{\pi}^B)$
9      else
10         $\theta = calculate\theta(\overline{\pi}^B, \pi^B, J^B)$
11         $\pi^B = \pi^B + \theta\overline{\pi}^B$
12 until $\sum_{e \in E} \overline{\pi}^B_e = 0$ ;
13 return $\overline{\pi}$
```

that in the R^{th} phase of the algorithm, a maximal set of plus edge disjoint augmenting paths of length i were augmented. In particular, an augmenting path from the unsaturated vertex $a \in U_1$ to the unsaturated $b \in U_2$ was augmented causing a particular edge $e = (u, v)$ to become saturated. This situation is illustrated in figure 4.2. Assume also that later in the S^{th} phase, an augmenting path $P_{c,d}$ of length $k > i$ from $c \in U_1$ through $\{x, v, u, y\} \in J^B$ to $d \in U_2$ was found. In this case, the augmentation process will cause edge e to become unsaturated. Finally assume in the $S + 1^{st}$ phase, an augmenting path $P_{f,g}$ of length $j < k$ from $f \in U_1$ through $\{x', u, v, y'\} \in J^B$ to $g \in U_2$ was found. Note that this is the only situation where augmenting path lengths in successive phases can decrease in length.

In the augmentations described above, the edges (x, v) and (u, y) are used as plus edges while the edges (x', u) and (v, y') are minus edges. Also note that $j > i$ since the lengths of the augmenting paths in phases 1 to R were assumed to be strictly increasing. Let $P_{(*),(**)}$ denote the path from vertex $(*)$ to vertex $(**)$ in figure 4.2, then we have:

$$k = |P_{c,v}| + |P_{u,d}| + 1$$
$$j = |P_{f,u}| + |P_{v,g}| + 1$$

Thus,

$$|P_{c,v}| + |P_{u,d}| + |P_{f,u}| + |P_{v,g}| < k + j$$

Now, consider the minimum of the two augmenting paths $P_{f,d}$ from f through $\{x', u, y\}$ to d and $P_{c,g}$ from c through $\{x, v, y'\}$ to g. Then we have:

$$\min\{|P_{c,v}| + |P_{v,g}|, |P_{f,u}| + |P_{u,d}|\} < \frac{k+j}{2} < k$$

We see that at the S^{th} phase, there was an augmenting path of length less than k which should have been found by the breadth first search. This proves the lemma by contradiction. □

Lemma 4.4.3. *In a given phase, let edge $e = (u, v)$ be any plus edge of an augmenting path P. After augmenting P, e cannot take part as a minus edge of another augmenting path P' in the same phase.*

Proof. Assume that in a given phase, there exists an edge $e = (u, v)$ that can be used both as a plus edge for the augmenting path $P = \{P_1 + \{(u, v)\} + P_2\}$ and a minus edge for the

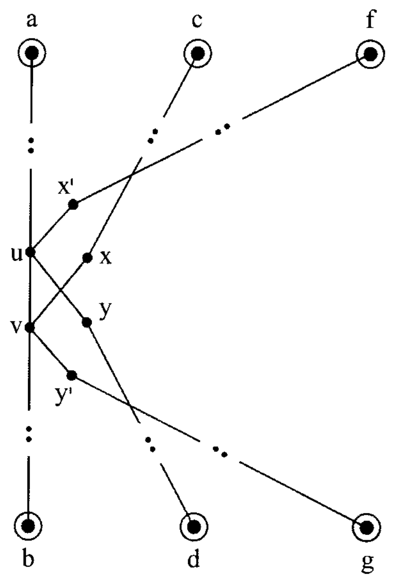


Figure 4.2: An example of an augmentation process

augmenting path $P' = \{P_1' + \{(v, u)\} + P_2'\}$, both of length k. Note that the two paths P and P' may also have other edges in common. Then

$$|P_1| + |P_2| + |P_1'| + |P_2'| < 2k$$

Furthermore, there must exist two augmenting paths $|P_1 + P_2'|$ and $|P_1' + P_2|$, the shorter of which has length

$$\min\left(|P_1 + P_2'|, |P_1' + P_2|\right) < k$$

Such a path cannot exist as the shortest augmenting path in this phase is of length k. The lemma follows by contradiction. □

Lemma 4.4.3 is extremely important to the running time of algorithm 4.5 as it allows us to safely delete plus edges in an augmenting path during a particular phase without fear of it returning later as a minus edge.

Lemma 4.4.4. *Given a non-optimal feasible solution $\overline{\pi}^B$ for an instance of* (DRP$_{\text{ufm}}$) *in G^B obtained by repeatedly finding augmenting paths and augmenting them from an initial*

DRP value of $\overline{\pi}^B = 0$, the largest set of augmenting paths $S = \{P_1, P_2, ..., P_k\}$ with respect to $\overline{\pi}^B$ have the properties that:

1. *$P_1, P_2, ..., P_k$ are plus edge disjoint.*
2. *after augmenting along these paths, we will arrive at the optimal solution $\overline{\pi}_{opt}$ for the instance of* ($\mathrm{DRP}_{\mathrm{ufm}}$).

Proof. We will first note that any given augmenting path P in a bipartite graph will never contain odd length cycles. Hence, after augmentation, all odd edges in P will increase its value by 1 whereas all even edges in P will decrease its value by 1.

Let $S = \{P_1, P_2, ..., P_k\}$ be the largest set of augmenting paths with respect to the current solution $\overline{\pi}^B$. Now, assume for the sake of contradiction that there are two augmenting paths P_i and P_j in S which share a plus edge e. According to ($\mathrm{DRP}_{\mathrm{ufm}}$), $\overline{\pi}_e^B \leq 1$, meaning that edge e can have value at most 1 and can only take part in one augmenting path. Hence, P_i and P_j cannot both exist in the set S. This shows that the set S must be plus edge disjoint.

Next, assume for the sake of contradiction that after augmenting along the paths in S that we have arrived at a new solution $\overline{\pi}^{B*}$ which is not an optimal solution. By theorem 4.3, there must exist an augmenting path P_{k+1} with respect to $\overline{\pi}^{B*}$. We have already shown that in order to increase the objective function, P_{k+1} must be plus edge disjoint from the set S, which contradicts the claim that $S = \{P_1, P_2, ..., P_k\}$ is the largest set of augmenting paths with respect to the original solution $\overline{\pi}^B$. Thus, the solution $\overline{\pi}^{B*}$ must be optimal. □

Lemma 4.4.4 tells us that given a feasible solution $\overline{\pi}$, there exists a set of plus edge disjoint augmenting paths that, when augmented, will result in an optimal solution $\overline{\pi}_{opt}$. This fact will become important as we analyze the running time of the phased algorithm.

4.4.3 Phased Algorithm Analysis

Theorem 4.4.5. *The phased algorithm in the previous section finds the optimal solution $\overline{\pi}_{opt}$ in $O(|V||E|)$ time.*

Proof. For each phase of the algorithm, a breadth first search tree is built by simultaneously spanning out from each of the unsaturated vertices in V_1 (with respect to π). The phase ends when an odd level containing unsaturated vertices in V_2 is found. The breadth first search tree can clearly be built in time $O(|E|)$ since no edge will be visited more than once.

The second part of the phase is the augmentation and topological plus edge delete. Notice that at the end of each iteration of the topological plus edge delete, if there are vertices in the set U_2 (unsaturated vertices in V_2) that have not been deleted, then there must be an augmenting path left in the BFS tree. Since the process of topological delete is proportional to the number of edges deleted. We conclude that this process can also be done in $O(|E|)$.

It is left to show that this algorithm will run $O(|V|)$ phases. From lemma 4.4.2, we have shown that the length of augmenting paths increase in each successive phase. The length of an augmenting path can be at most $|V|$. Therefore, we conclude the number of phases is bounded by $O(|V|)$. The theorem follows. □

Using the phased algorithm to solve ($\mathrm{DRP}_{\mathrm{ufm}}$) in place of **DRPopt**$(J)$ in algorithm 4.2 therefore will result in a running time of $O(|V|^2|E|)$.

Chapter 5

Conclusion

In this thesis, we presented two combinatorial algorithms to solve the unconstrained fractional matching problem. The naïve algorithm runs in $O(|V||E|^2)$ time. The "phased algorithm" runs in $O(|V|^2|E|)$, a significant improvement over the naïve algorithm. The unconstrained fractional matching problem is the first step in our attempt to design a combinatorial algorithm for the maximum charge problem. Our representation of the problem allows vertex capacities but not edge capacities as edge values π are unconstrained. We now consider the maximum charge problem.

Given a graph $G(V, E)$ the primal variable vector is made up of variables representing vertices $v \in V$, edges $e \in E$, as well as slack varaibles ϵ_e for each edge $e \in E$. The coefficients c_u (c_e) are capacities given to the vertices (edges) of G. The primal problem is represented by

$$\begin{array}{lll} \textit{minimize} & \sum_{u \in V} c_u x_u + \sum_{e \in E} c_e x_e & \\ \textit{subject to} & x_u + x_v + x_e - \epsilon_e = 1 & \forall (u, v) \in E \\ & x_u \geq 0 & \forall u \in V \\ & x_e \geq 0 & \forall e \in E \\ & \epsilon_e \geq 0 & \forall e \in E \end{array} \tag{$\mathrm{P_{mc}}$}$$

The maximum charge problem, which is represented by the dual representation, is as follows:

$$\begin{aligned}
&\textit{maximize} && \sum_{e\in E} \pi_e && && (\mathrm{D}_{\mathrm{mc}})\\
&\textit{subject to} && \sum_{e\in E(u)} \pi_e \le c_u && \forall u \in V\\
& && \pi_e \le c_e && \forall e \in E\\
& && \pi_e \ge 0 && \forall e \in E\\
& && \pi_e \gtrless 0 && \forall e \in E
\end{aligned}$$

We now define the sets J_1, J_2, and J_3 as

$$J_1 = \left\{ u : \sum_{e\in E(u)} \pi_e = c_u \right\} \tag{5.1}$$

$$J_2 = \{e : \pi_e = c_e\} \tag{5.2}$$

$$J_3 = \{e : \pi_e = 0\} \tag{5.3}$$

From here, we derive the RP and DRP for the maximum charge problem.

$$\begin{aligned}
&\textit{minimize} && \sum_{e\in E} s_e && && (\mathrm{RP}_{\mathrm{mc}})\\
&\textit{subject to} && x_u + x_v + x_e - \epsilon_e + s_e = 1 && \forall (u,v) \in E\\
& && s_e \ge 0 && \forall e \in E\\
& && x_u \ge 0 && \forall u \in J_1\\
& && x_u = 0 && \forall u \notin J_1\\
& && x_e \ge 0 && \forall e \in J_2\\
& && x_e = 0 && \forall e \notin J_2\\
& && x_e \ge 0 && \forall e \in J_3\\
& && x_e = 0 && \forall e \notin J_3
\end{aligned}$$

$$\begin{array}{llll} \textit{maximize} & \displaystyle\sum_{e\in E} \pi_e & & \\ \textit{subject to} & \displaystyle\sum_{e\in E(u)} \pi_e \leq 0 & \forall u \in J_1 & \\ & \pi_e \leq 0 & \forall e \in J_2 & \\ & \pi_e \geq 0 & \forall e \in J_3 & \\ & \pi_e \gtrless 0 & \forall e \in E & \\ & \pi_e \leq 1 & \forall e \in E & \end{array} \tag{DRP$_{\text{mc}}$}$$

It is immediately clear that a similar combinatorial algorithm for the DRP of the maximum charge problem can be devised by repeatedly finding augmenting paths (with respect to sets J_1, J_2, and J_3) corresponding to equation DRP$_{\text{mc}}$ and augmenting them. However, it is not clear how many times this DRP algorithm will run. After each iteration, an optimal solution $\overline{\pi}_{opt}$ to (DRP$_{\text{mc}}$) will be found. This solution will either cause an unsaturated vertex (or edge) value to be saturated, or an edge value to go to 0. It is not immediately evident whether edges entering the sets J_2 and J_3 will come out of the set at a later time. Thus, this is an interesting topic for future research.

Another possible topic is given an optimal solution to the unconstrained fractional matching problem, obtain an optimal solution to the maximum charge problem. This may be done through a series of de-augmentations of negative edge values. As a simple example, the graph on the right hand side of figure A.9 in appendix A.3 has two de-augmenting paths represented by the paths {(c,d),(d,e),(e,f)} and {(g,d),(d,e),(e,h)}. After the de-augmentation, which involves subtracting 1 from the plus edges and adding 1 to the minus edges, the new solution will resemble those of figure A.7 and A.8.

Appendix A

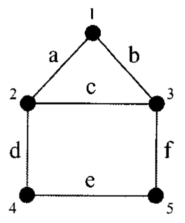
Matching Examples

A.1 Fractional Matching Formulation

The following is a fractional matching LP representation of the graph in figure A.1:

$$\begin{array}{ll}
\textit{maximize} & \pi_a + \pi_b + \pi_c + \pi_d + \pi_e + \pi_f \\
\textit{subject to} & \pi_a + \pi_b \leq 1 \\
 & \pi_a + \pi_c + \pi_d \leq 1 \\
 & \pi_b + \pi_c + \pi_f \leq 1 \\
 & \pi_d + \pi_e \leq 1 \\
 & \pi_e + \pi_f \leq 1 \\
 & \pi_a, \pi_b, \pi_c, \pi_d, \pi_e. \pi_f \geq 0
\end{array}$$

The optimal solution for this graph is shown in figure A.8 in section A.3.



$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
a & 1 & 1 & 0 & 0 & 0 \\
b & 1 & 0 & 1 & 0 & 0 \\
c & 0 & 1 & 1 & 0 & 0 \\
d & 0 & 1 & 0 & 1 & 0 \\
e & 0 & 0 & 0 & 1 & 1 \\
f & 0 & 0 & 1 & 0 & 1
\end{array}$$

Figure A.1: A sample graph and the corresponding incidence matrix A

A.2 Shortest Path Primal-Dual Example

In this section, we will walk through the primal-dual interpretation of the shortest path problem as described in section 3.3.1. Figure A.2 shows a typical problem of this type.

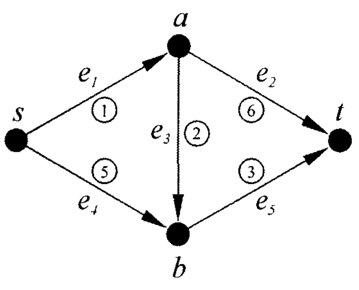


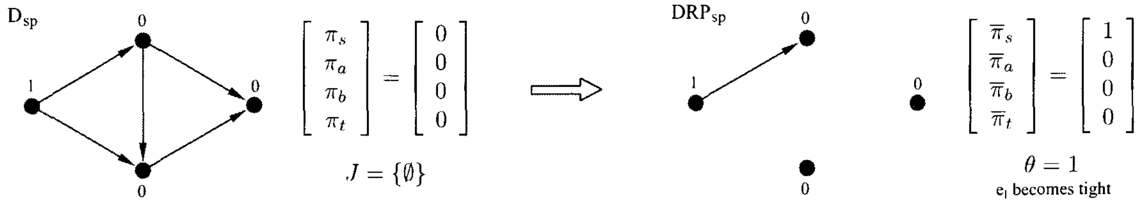
Figure A.2: Shortest path problem with edge weights indicated in the circles

The corresponding 4×5 edge-vertex incidence matrix A is as follows:

$$A = \begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline s & +1 & 0 & 0 & +1 & 0 \\ a & -1 & +1 & +1 & 0 & 0 \\ b & 0 & 0 & -1 & -1 & +1 \\ t & 0 & -1 & 0 & 0 & -1 \end{array}$$

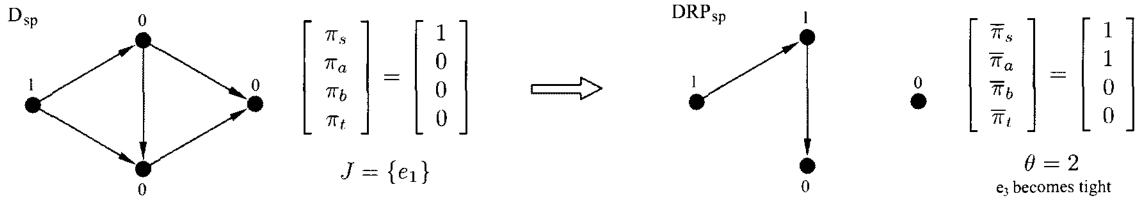
Unlike the previous example, the columns of A represent the edges and rows of A represent the vertices. This is because fractional matching is a maximization problem modeled from the dual's point of view whereas the shortest path is a minimization problem seen from the primal's point of view.

Figures A.3 to A.6 show the progression of four iterations of the shortest path primal-dual algorithm. The algorithm begins with an initial dual solution of $\pi = [0, 0, 0, 0]$ and through successive improvements, arrives at the optimal solution $\pi = [6, 5, 3, 0]$. The algorithm terminates in the state shown in figure A.6; the corresponding shortest path is the set $J = [e_1, e_3, e_5]$.



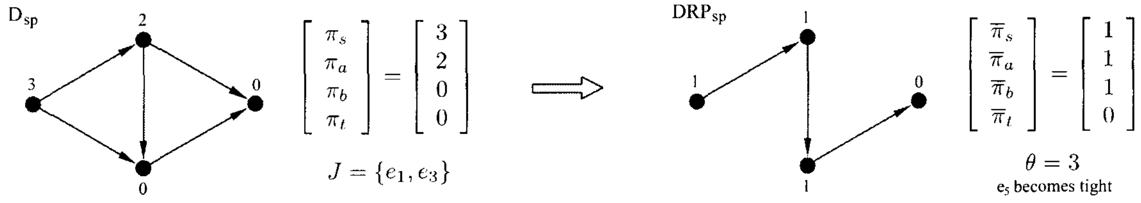
$$\begin{bmatrix}\pi_s\\ \pi_a\\ \pi_b\\ \pi_t\end{bmatrix} = \begin{bmatrix}0\\0\\0\\0\end{bmatrix} \qquad J = \{\emptyset\} \qquad \Longrightarrow \qquad \begin{bmatrix}\overline{\pi}_s\\ \overline{\pi}_a\\ \overline{\pi}_b\\ \overline{\pi}_t\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

Figure A.3: Iteration 1, $(\overline{\pi}_{opt})_s - (\overline{\pi}_{opt})_t = 1$



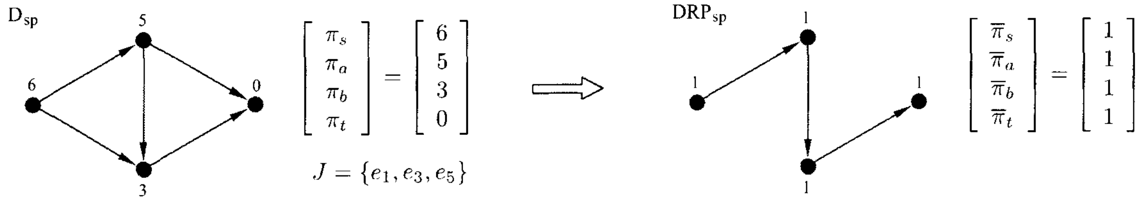
$$\begin{bmatrix}\pi_s\\ \pi_a\\ \pi_b\\ \pi_t\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad J = \{e_1\} \qquad \Longrightarrow \qquad \begin{bmatrix}\overline{\pi}_s\\ \overline{\pi}_a\\ \overline{\pi}_b\\ \overline{\pi}_t\end{bmatrix} = \begin{bmatrix}1\\1\\0\\0\end{bmatrix}$$

Figure A.4: Iteration 2, $(\overline{\pi}_{opt})_s - (\overline{\pi}_{opt})_t = 1$



$$\begin{bmatrix}\pi_s\\ \pi_a\\ \pi_b\\ \pi_t\end{bmatrix} = \begin{bmatrix}3\\2\\0\\0\end{bmatrix} \qquad J = \{e_1, e_3\} \qquad \Longrightarrow \qquad \begin{bmatrix}\overline{\pi}_s\\ \overline{\pi}_a\\ \overline{\pi}_b\\ \overline{\pi}_t\end{bmatrix} = \begin{bmatrix}1\\1\\1\\0\end{bmatrix}$$

Figure A.5: Iteration 3, $(\overline{\pi}_{opt})_s - (\overline{\pi}_{opt})_t = 1$



$$\begin{bmatrix}\pi_s\\ \pi_a\\ \pi_b\\ \pi_t\end{bmatrix} = \begin{bmatrix}6\\5\\3\\0\end{bmatrix} \qquad J = \{e_1, e_3, e_5\} \qquad \Longrightarrow \qquad \begin{bmatrix}\overline{\pi}_s\\ \overline{\pi}_a\\ \overline{\pi}_b\\ \overline{\pi}_t\end{bmatrix} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

Figure A.6: Iteration 4, $(\overline{\pi}_{opt})_s - (\overline{\pi}_{opt})_t = 0$

A.3 Optimal Solutions

Examples illustrated in figures A.7 to A.9 show the differences between optimal solutions to the 0-1, fractional, and unconstrained fractional matching problems. Note that some optimal solutions shown here are not unique. The left is a general graph while the right is bipartite.

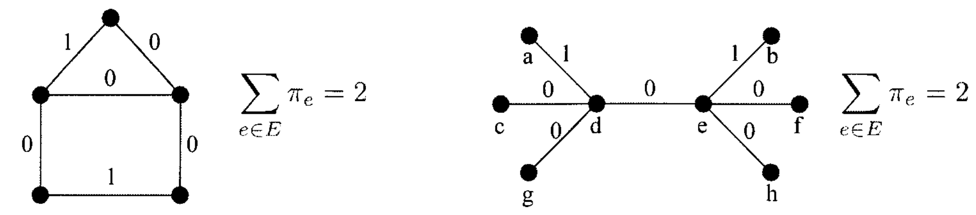


Figure A.7: 0-1 matching, $\pi_e = \{0, 1\}$

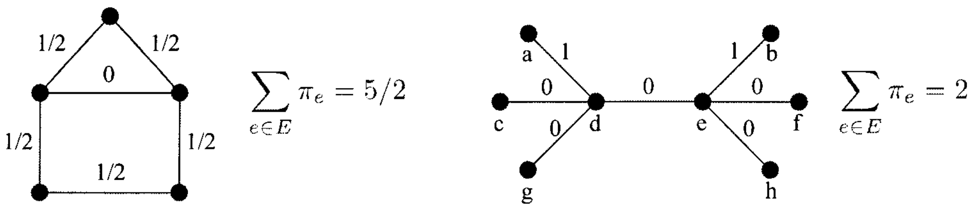


Figure A.8: fraction matching, $\pi_e \geq 0$

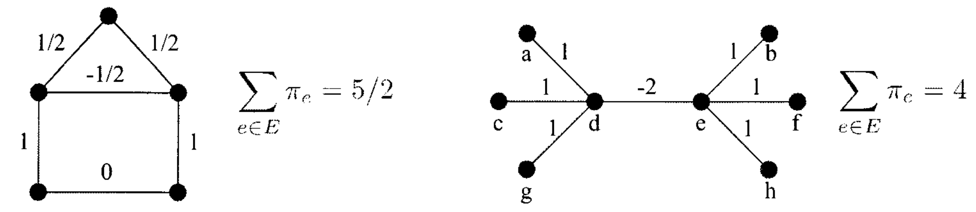


Figure A.9: unconstrained fractional matching (vertex capacities $c(u) = 1$), $\pi_e \gtrless 0$

Bibliography

[AMO93] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flow: Theory, Algorithms, and Applications.* Englewood Cliffs, N.J.: Prentice Hall, 1993.

[Ber57] C. Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences*, 43:842–844, 1957.

[BP89] J.M. Bourjolly and W.R. Pulleyblank. König-Egerváry graphs, 2-bicritical graphs and fractional matchings. *Discrete Applied Mathematics*, 24:63–82, 1989.

[Chv83] V. Chvátal. *Linear Programming.* W.H. Freeman and Company, 1983.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition).* Massachusetts Institute of Technology, 2001.

[Dan63] G.B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, 1963.

[DLF56] G.B. Dantzig and D.R. Fulkerson L.R. Ford. A primal-dual algorithm for linear programs. *Linear Inequalities and Related Systems*, pages 171–181, 1956.

[Edm65a] J. Edmonds. Maximum matching and a polyhedron wih 0-1 vertices. *Journal of research of the national bureau of standards – B. Mathematics and Mathematical Physics*, 69B(1 and 2):125–130, Jan - June 1965.

[Edm65b] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–469, 1965.

[Ege31] E. Egerváry. On combinatorial properties of matrices. *Matematikai Lapok*, 38:16–28, 1931.

[EK72] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association for Computing Machinery*, 19(2):248–264, 1972.

[FF58] L.R. Ford and D.R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, 1958.

[HK73] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal of Computing*, 2(4):225–231, 1973.

[K̈31] D. König. Graphen und matrizen. *Matematikai Lapok*, 38:116–119, 1931.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[KGGS06] R. Krishnamurti, D.R. Gaur, S.K. Ghosh, and H. Sachs. Berge's theorem for the maximum charge problem. *Discrete Optimization*, 3(2):174–178, 2006.

[Kha79] L.G. Khachiyan. A polynomial time algorithm for linear programming. *Soviet Mathematics Doklady*, 20:1092–1096, 1979.

[KM72] V. Klee and G.J Minty. How good is the simplex algorithm? *Inequalities III*, pages 159–175, 1972.

[KM74] T. Kameda and I. Munro. An $O(|V||E|)$ algorithm for maximum matching of graphs. *Computing*, 12(1):91–98, March 1974.

[Koz92] D. Kozen. *The design and Analysis of Algorithms.* Springer, 1992.

[Kuh55] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[MPKM78] V. M. Malhotra, M. Pramodh-Kumar, and S. N. Maheshwari. An $O(|V|^3)$ algorithm for finding maximum flows in networks. *Information Processing Letters*, 7(1):277–278, January 1978.

[MS04] M. Mucha and Piotr Sankowski. Maximum matchings via gaussian elimination. *Proceedings. 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004.

[PS82] C.H. Papadimitriou and Kenneth Steiglitz. *Combinational Optimization: Algorithms and Complexity.* Dover Publications Inc., 1982.

[Riz00] R. Rizzi. A short proof of könig's matching theorem. *Journal of Graph Theory*, 33(3):138–139, March 2000.

[SU97] E. Scheinerman and D.H. Ullman. *Fractional Graph Theory: A Rational Approach to the Theory of Graphs.* John Wiley & Sons, Inc., 1997.

[Vaz94] V. Vazirani. A theory of alternating paths and blossoms for proving correctness of the $O(\sqrt{|V|}|E|)$ general graph matching algorithm. *Combinatorica*, 14(1):71–109, 1994.