Dinitz Problem

Kechun (Coco) Mao, Tianlin Liu
kmao@andrew.cmu.edu, t.liu@jacobs-university.de

Mathematics Department
Jacobs University University
Campus Ring, Bremen, Gremany

Abstract

In this paper, we basically follow the chapter Dinitz Problem of Aigner and Zieglers' book Proof from THE BOOK, 4th edition. Our main contribution was to make the material more readable and more beginner-friendly. We introduce Dinitz Problem, its background, and its full proof in this paper. We will assume basic knowledge of graph thoery from the readers and thus do not give definitions of some basic graph theory concepts, but we will give many clear definitions and abundant examples to help readers understand the material.

1. Introduction

Jeff Dinitz proposed a simple-sounding problem in 1978, but it was not answered until 1994 by Fred Galvin with a surprisingly simple solution. Dinitz problem is as follows:

Consider n^2 cells arranged in an $(n \times n)$-square, and let (i, j) denote the cell in row i and column j. Suppose that for every cell (i, j) we are given a set $C(i, j)$ of n colours. Is it then always possible to colour the whole array by picking for each cell (i, j) a colour from its set $C(i, j)$ such that the colours in each row and each column are distinct?

Such a square where the colour in each row and column are chosen from a "local" set of colours and every colours in one row or one column are distinct is called **Partial Latin Square**[2].

A special case is when all colour sets $C(i, j)$ are the same and the square is called **Latin Suqare**. For Latin Square, it was proved in 1981 that if fewer than n cells are filled in an $(n \times n)$-array, one can always complete it to obtain a Latin square (Evans conjecture). However, the difficulty of Partial Latin Square derives from the fact that not every colour of C is available at each cell. For example, for the following square

{1,2}	{2,3}
{1,3}	{2,3}

While there indeed exists partial Latin square, that is

2	3
1	2

If we happen to choose 1 and 2 for the first row, we will not find valid numbers to fill in for the second row; and the other hand, if we are lucky enough to choose 2 and 3 for the first row, then we legal to pick 3 and 2 in the second row and we are done. The question is, without trial and error, how shall we see whether this graph can be coloured properly immediately or not? Further, what is the condition that guarantees us to have a proper colouring? This is what the Dinitz problem concerns about.

We will rephrase Dinitz problem in the language of graph theory to facilitate the proof later. To do so, we will introduce following definitions

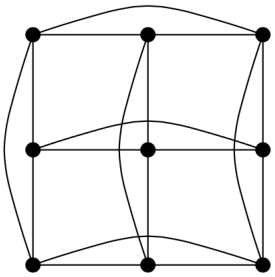
Definition. *A **proper colouring** of a graph is an assignment of colours to the vertices of the graph so that no two adjacent vertices have the same colour.*

Definition. *A **List colouring** is a proper colouring c such that c: $V \longrightarrow \cup_{v \in V} C(v)$, where $c(v) \in C(v)$, $\forall v \in V$*

Definition. ***List chromatic number** of a graph G, denoted as $\chi_l(G)$, is the smallest number k such that **any** list of colour sets $c(v)$, $|c(v)| = k$, $\forall v \in V$ always exits a list colouring.*

We will also need to turn the square into a graph, (V, E). For a n×n square, we will construct a S_n as follows:
Each cell is represented by a vertex. There is an edge

between two vertices if the cells they represent are in the same row or the same column.
For 3×3 square, S_3 is as follows



Since any n cells in a row are pairwise adjacent, we need at least n colours, that is $X_l(S_n) \geq n$.

Finally, we can summarize the Dinitz problem into one simple question:

$$X_l(S_n) = n?$$

This expression is succint and accurate.
We shall see in the following sections how we can prove $X_l(S_n) = n?$.

2 Lemmas

Fred Galvin, who solved Dinitz problem, combined two well-known math results. We shall introduce and prove these two math results that Galvin used before diving into the main proof.

2.1 Kernel and Directed Graph

In order to study Dinitz Problem rigorously, we need to introduce the notion of kernel in directed graph. Then, we will give out our lemma 1 which turned out to be the key for this problem.

Definition. *Let $\vec{G}$ be a directed graph. A **kernel** $K \subseteq V$ is a subset of the vertices such that:*

1. *K is independent in G*
2. *For every $u \notin K$ there exists a vertex $v \in K$ with an edge $u \longrightarrow v$*

Definition. *For a subset A $\subseteq$ V we denote by G_A the subgraph which has A as vertex set and which contains all edges of G between vertices of A. We say G_A the subgraph **induced** by A.*

Definition. *For a vertex v $\in V$, the outdegree $d^-(v)$ is the number of edges with v as initial vetex, and indegree $d^+(v)$ is the number of edges with v as the target vertex.*

Lemma 1. *Let $\vec{G} = (V, E)$ be a directed graph with each vertex v having a colour set $C(v)$, if both conditions below are satisfied:*

1. *$|C(v)| \leq d^+(v) + 1$*
2. *every induced graph of $\vec{G}$ has a kernel*

then there exists a list colouring of G with a colour from C(v) for each v.

Proof.

We will prove by induction on the number of vertices $|V|$.

Base Case: When $|V|$ = 1, the statement is trivially correct, since we can just assign the only colour we have to that single vertex.

Induction Hypothesis: We inductively suppose that for $|V| = n$ and it satisfied the two conditions above, then there exists a list colouring of G with a colour from $C(v)$ for each v.

Step Case: When $|V| = n + 1$, we need to show that there exists a list colouring of G with a colour from $C(v)$ for each v.

Pick any colour $c \in C = \cup_{v \in V} C(v)$. Define a set $A(c) := v \in V : c \in C(v)$

Since A is a subset of V, the graph A induced processes a kernel, say $K(c)$. Recall that a kernel is independent, and therefore there is no edges connecting any 2 vertices in this kernel. Hence it is legal to colour all these vertices in $K(c)$ with the same colour c.

Now, we delete c from any $C(v)$ that c was in and delete $K(c)$ from G. That is to say, we give up using the colour c, and at the same time we erase all vertices that are in the colour c. Let's call this new graph G', call the new colour set C'

Next, let's have a look at the new graph G' induced by $V \setminus K(c)$. Our goal is to show that this new graph falls into our Induction Hypothesis case.

Clearly, as we deleted at least one vertex from the graph G, now $|V| \leq n$. We are left to show that G' satisfies the two conditions listed in the lemma statement. Note that the second condition obviously holds, because any subgraph in G' is a subgraph in G. The more complicated part is to

verify $|C(v)| \leq d^+(v) + 1$ for any v.

Pick any vertex $v \in A(c) \setminus K(c)$. Because of the second condition of kernel, before the deleting process, there was an edge went out of v and connects a vertex $u \in K(c)$. That is to say, after the deleting, d^+v decreased at least by 1. Since $|C(v)|$ decreases by 1 exactly and d^+v decrease by at least 1, the condition $|C(v)| \leq d^+(v) + 1$ is still satisfied. Note that if v is out of $A(c)$, then $C(v)$ does not change and $d^+(v)$ can only go down, therefore it satisfied the condition, too.

Now, we can use our induction hypothesis on G', that is G' has a list colouring. Then if we combine G''s list colouring with the deleted vertices colored in c, we will arrive at a list coloring for G with a colour from C(v) for each v.

□

2.2 Stable Matching Theorem

The second lemma will require following definitions.

Definition. *A **bipartite graph** $G = (X \cup Y, E)$ is a graph with the following property: The vertex set V is split into two disjoint parts X and Y such that every edge has one endvertex in X and the other in Y.*

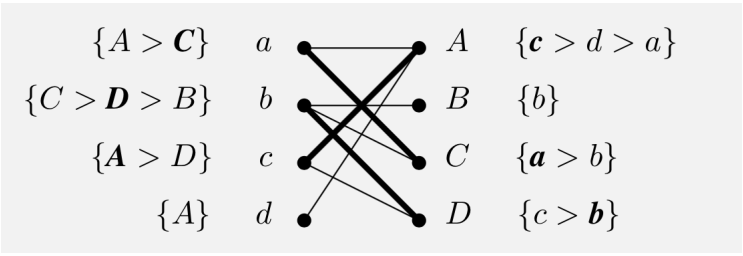
Definition. *A **matching** M in a bipartite graph G is a set of edges such that no two edges in M have a common endvertex.*

Definition. *In $G = (X \cup Y, E)$, a **ranking** of vertices adjacent to a vertex v, denoted by **N(v)** (neighbors of v), is a list such that $\{z_1 > z_2 > \ > z_d(v)\}$, in which z's are listed in the descending order by v's preferences.*

Definition. *A **matching** M of $G = (X \cup Y, E)$, in which each vertex has a ranking of its adjacent vertices, is called stable if the following condition holds: Whenever $uv \in E \setminus M$, $u \in X$, $v \in Y$, then either $uy \in M$ with $y > v$ in $N(u)$ or $xv \in M$ with $x > u$ in $N(v)$, or both.*

To understand these concepts in a more intuitive way, we shall give the following analogy in real life. For bipartite graph, we can imagine a group of boys to be vertices in X and a group of girls to be vertices in Y (not necessairly equal number). If a guy and a girl have potential to marry, there is an edge between them. For a girl, she may prefer one guy over another, so each person has a ranking of the opposite gender. A matching is then a mass-wedding with no person committing bigamy. Finally, a matching is stable if it never happens that u and v are not married but u prefers v to his partner (if he has one at all) and v prefers u to her mate (if she has one at all).

The following graph shows a bipartite graph that has a stable matching (matching edges in bold).



Our second theorem claims that

Lemma 2. *A stable matching always exists for any bipartite graph of any possible ranking of $N(v)$ for each v.*

Proof.
We will give an algorithm and show that running the algorithm indeed finds a stable matching. For ease of understanding, we will describe the algorithm in the language of the real life example.
The algorithm to find a stable matching of couples to marry runs as follows. We will have a set R and $R = X$ at first, that is, R contains all the men.

1. All men in R propose to their top choice among girls they have not been rejected yet.
2. If a girl receives multiple proposals, she keeps the one she likes the best and keeps him on the strings. If she already has a man on the string from last round, she considers him together with the new proposals and keeps the best on the string. She rejects all other men.
3. The men who gets rejected form a reservoir R. However, a man who proposes his last choice and gets rejected is out of our consideration, that is, he will not get a match. If R is empty, we stop and the couples on the strings form a stable matching. If R is not empty yet, go back to step 1.

We claim that those couples on the strings indeed form a stable matching. The argument is in fact quite straightforward by checking the definition of *stable matching*.
If $uv \in E$ but $uv \notin M$, then one of the following cases must have happened

1. If u did not propose to v ever, it must be the case that u, the man, found a better girl and got accepted before he even got around to v. This case implies $\exists uy \in M$ with $y > v$ in $N(u)$.
2. If u indeed proposed to v before but got rejected, it must be the case that v, the girl, rejected u because another man she likes more proposed to her. This case implies $\exists xv \in M$ with $x > u$ in $N(v)$.

This is exactly the condition of a stable matching. □

3 The Solution to Dinitz Problem

The answer to Dinitz Problem is positive, that is $X_l(S_n) = n$ indeed holds.

Before we actually start to prove it, let's see how shall we proceed our proof in order to fully use our lemmas. If we can form a directed graph from the $n \times n$ square with such an orientation that each vertex v satifies the condition $d^+(v) \leq |C(v)| - 1 = n - 1$ and any subgraph of this directed graph has a kernel, by lemma 1, we will be done. To show there is indeed a kernel for every subgraph, we "fish" our kernel by using the property of stable matching. That is to say, we need to verify three things:

1. We can construct a directed graph $\vec{G}$ from S_n
2. $d^+(v) \leq |C(v)| - 1 = n - 1$
3. Every subgraph of $\vec{G}$ possesses a kernel.

We will verify above three conditions step by step. Please note that in **Step 1**, we verify (1); in **Step 2**, we verify (2); in **Step 3** and **Step 4**, we verify (3). After these 4 steps, we simply use lemma 1, which directly provides our desired result.

Step 1 (Construct a directed graph):

We denote the vertices of S_n by

$$(i, j), 1 \leq i, j \leq n$$

Thus (i, j) and (r, s) are adjacent if and only if $i = r$ or $j = s$.

Let L be a latin square. Denote the entry of this latin square at (i, j) position by $L(i, j)$. Next, we make S_n into directed graph $\vec{S_n}$.

We denote:

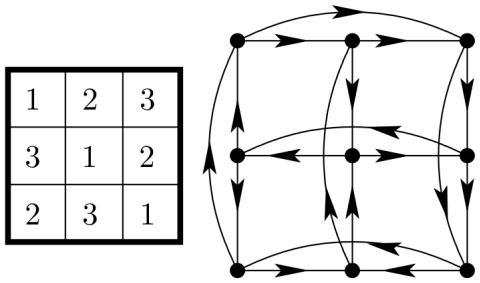
$$(i, j) \longrightarrow (i, j') \text{ if and only if } L(i, j) < L(i, j')$$

Moreover, We denote:

$$(i, j) \longrightarrow (i', j) \text{ if and only if } L(i, j) > L(i', j)$$

That is to say, on a row, we point smaller entries to bigger entries; on a column, we point bigger entries to smaller entries. In this way, we finish our directed graph constructing for the step 1.

The following is an example of a directed graph



Step 2 :

Claim: $d^+(i, j) = n - 1, \forall, (i, j)$.

This claim is more or less obvious to verify. Consider any $(i, j) = k$ in S_n. On a row, there are $n - k$ entries that are bigger than k, so there are $n-k$ edges going out of (i, j) in the row; on the other hand, there are $k - 1$ entries that are smaller than k in a column. That is to say, there are $k - 1$ edges pointing out of (i, j). A simple computation gives us:

$$d^+(i, j) = n - k + k - 1 = n - 1$$

In the following steps, our goal is to show that every subgraph of $\vec{G}$ possesses a kernel. As mentioned in the beginning of this section, to find a kernel for every subgraph, we will use the stable matching property of bipartite in lemma 2. So before that, let's construct a bipartite for each subgraph.

Step 3 (Construct a bipartite for each subgraph):

Pick any subset $A \subseteq V$, let X be the set of rows of L, and Y the set of its columns. We than can construct a bipartite graph $G = (X \cup Y, A)$ in the following way: for every $(i, j) \in A$, draw an edge linking $i \in X$, $j \in Y$.

Now, in order to use Lemma 2, we need to introduce a ranking on the bipartite graph $G = (X \cup Y, A)$. Let $j' > j$ in N(i) (remind that N(i) is the "choices" for i, in a decreasing order of favour), if $(i, j) \longrightarrow (i, j')$. Let $i' > i$ in N(j), if $(i, j) \longrightarrow (i', j)$. Now, $G = (X \cup Y, A)$ is a bipartite with ranking. Then by Lemma 2, it has a stable matching M.

Step 4 :

Claim: Every subgraph of $\vec{G}$ possesses a kernel. Specifically, M in step 3 is a kernel for A, for any A as a subset of G.

To prove M is a kernel for A, firstly let's prove that M is independent in A. This is quite obvious because for

a stable matching, there is only one edge from i to j, for (i, j) fixed in M. This means, there is no edges of M in $G = (X \cup Y, A)$ sharing a same vertex.

Next, assume $(i, j) \in A \setminus M$. Then by definition of stable matching, either exists $(i, j') \in M$ with $j' > j$, or $(i', j) \in M$ with $i' > i$, or both. For $\vec{S_n}$, this means $(i, j) \longrightarrow (i, j') \in M$ or $(i, j) \longrightarrow (i', j) \in M$. Hence M is indeed a kernel.

So far, we checked all the conditions for using Lemma 1, which guarantees that n is indeed the smallest colouring set that we can give all vertices in rows/columns distinctive colours. Hence we are done.

4 Conclusion

In this paper we present the proof for the Dinitz Problem in a piecemeal manner. In the section 1, we introduced basic definitions to set up our question in a graph theory fashion. In the section 2, we introduced two lemmas which turned out crucial for solving the Dinitz Problem. In section 3, we solve the Dinitz problem with the help of two lemmas.

5 Acknowledgement

The authors would like to gratefully and sincerely thank Dr. Marcel Oliver for his instruction and guidance in and out of the class *Perspective of Mathematics Spring 2015*.

The authors would also like to thank all students from the class *Perspective of Mathematics Spring 2015*, Denis Igorevich Korolev, Ekber Shahkeremov, Emanuel Stiuler, Remus Marius Dumitrel. Thank them for their valuable comments, suggestions, and assistances.

Please note that all figures in this paper are credited to *The Proof from THE BOOK*.

6 References

[1] M. Aigner, G.M. Ziegler, *Proofs from THE BOOK*, Springer, 2003.

[2] Partial Latin Square. (n.d.). Retrieved May 10, 2015, from http://mathworld.wolfram.com/PartialLatinSquare.html