Perfect Matchings After Vertex Deletions in n-dimensional Lattice Graphs

by

Hangjun Yang

B.Sc., Zhejiang University, China, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES

Mathematics

The University of British Columbia

August 2005

Abstract

This thesis studies lattice graphs which are readily seen to have many perfect matchings and considers whether if we delete vertices the resulting graphs continue to have perfect matchings. It is clear that one can destroy the property of having a perfect matching by deleting an odd number of vertices, by deleting all the neighbours of a given vertex, etc. Besides these trivial "destructions", in order to guarantee the resulting graph still have perfect matchings, we require the deleted vertices to be mutually far apart. In this thesis, we consider an n-dimensional lattice graph $Q(m, n)$ with bipartition of black and white vertices, where m is even. If the distance of any two deleted black (or white) vertices is greater than $4n(n+1)\sqrt{m}$, then the resulting graph (after vertex deletions) continues to have a perfect matching.

Contents

Abstract ii

Contents iii

List of Tables v

List of Figures vi

Acknowledgements vii

1 Background 1

1.1 Basic Concepts 1

1.2 Matchings 3

1.2.1 Matchings in Bipartite Graphs 3

1.2.2 Matchings in General Graphs 5

1.3 Networks and Flows 6

1.4 Capacitated Perfect b-matchings and Bidirected Flows 8

2 Applications of Matchings 12

2.1 Applications of Matchings in Bipartite Graphs 12

2.1.1 The Assignment Problem 12

2.1.2 The Dimer Problem 14
2.2 Applications of Matchings in General Graphs 16
2.2.1 The Chinese Postman Problem 16
2.2.2 Quadrilateral Mesh Generation in Computer-Aided Design . 17

3 Introduction of the Problem **19**
3.1 Robustness 19
3.2 Motivation 20
3.3 Definitions and Notations 21
3.4 A Result in the 2-D Case 23

4 Main Result for n-dimensional Lattice Graphs **25**
4.1 Statement of Theorem 4.1 25
4.2 Proof of Theorem 4.1 29

Bibliography **36**

List of Tables

2.1 Tableau Form of the Kabuki Electronics Assignment Problem 13

2.2 The Optimal Assignment of the Kabuki Electronics Assignment Problem 14

List of Figures

1.1 A pictorial representation of a graph 2

1.2 A matching M_1 and a perfect matching M_2. 3

3.1 $Q(m, 2)$ gives an $m \times m$ lattice graph, here $m = 20$. 22

4.1 A 2-D diamond with $d = 8$ in the checkerboard with $m = 20$. 27

4.2 A sketch of a 3-D diamond. 28

Acknowledgements

This thesis was conducted under the supervision of Dr. Richard Anstee. I would like to thank him for his guidance, advice and encouragement.

Hangjun Yang

The University of British Columbia
August 2005

Chapter 1

Background

This chapter mostly follows the terminology in the textbook *Introduction to Graph Theory* by West [31].

1.1 Basic Concepts

When speaking of graphs we might think about pictures of some kind that are used to represent information such as bar graphs, flow charts and graphs of functions. But the graphs we shall discuss are quite different. They correspond more closely with the objects which, in everyday language that we call "networks".

A *graph* G with n *vertices* and m *edges* consists of a *vertex set* $V(G) = \{v_1, \ldots, v_n\}$ and an *edge set* $E(G) = \{e_1, \ldots, e_m\}$, where each edge consists of two (possibly equal) vertices in $V(G)$ called its *endpoints*. We usually write $G = (V(G), E(G))$, or simply $G = (V, E)$. A *directed graph* or *digraph* G consists of a vertex set $V(G)$ and an edge set $E(G)$, where each edge is an ordered pair of vertices.

A *loop* is an edge whose endpoints are equal. *Parallel edges* or *multiple edges* are edges that have the same pair of endpoints. A *simple graph* is a graph having

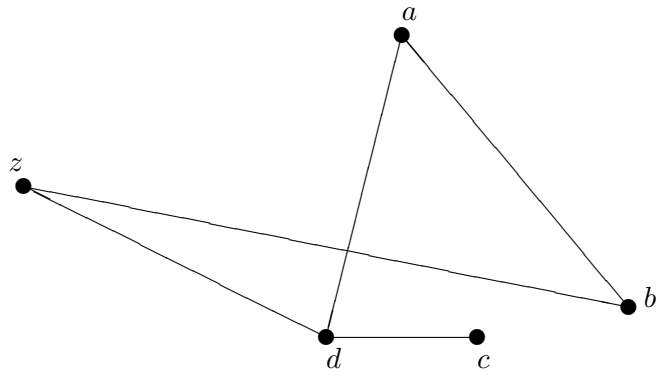

Figure 1.1: A pictorial representation of a graph

no loops or multiple edges. A *complete graph* is a simple graph in which every pair of vertices is an edge. Denote a complete graph with n vertices by K_n.

A graph is finite if its vertex set and edge set are finite. We emphasize here that every graph mentioned in this paper is finite.

A typical example of a simple graph $G = (V, E)$ is given by the sets

$$V = \{a, b, c, d, z\}, E = \{(a, b), (a, d), (b, z), (c, d), (d, z)\}.$$

We represent the vertices as points, and join two points by a line whenever the corresponding pair of vertices is an edge. Thus Figure 1.1 is a pictorial (or graphical) representation of the graph given in the example above.

A *walk of length* k is a sequence $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$ of vertices and edges such that $e_i = v_{i-1}v_i$ for all i. A *path* is a walk with no repeated vertex. A *trail* is a walk with no repeated edge. A walk (or trail) is *closed* if it has length at least one and its endpoints are equal. A walk $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$ whose vertices are all distinct except that $v_0 = v_k$ is called a *cycle*. We call a cycle *odd* if it has an odd number of distinct vertices (and an odd number of edges).

A walk specifies a route in G which proceeds through a sequence of vertices joined by edges. A walk may visit any vertex an arbitrary number of times. In a path, each vertex is visited at most once. A *u,v-path* is a path whose first vertex is

u and last vertex is v.

A graph G is *bipartite* if $V(G)$ is the union of two disjoint sets X and Y such that each edge consists of one vertex from X and one vertex from Y. We say G is a bipartite graph with bipartition X, Y.

Here is an important fact about bipartite graphs.

Theorem 1.1 *A graph G is bipartite if and only if it contains no odd cycles.*

Theorem 1.1 implies that whenever a graph G is not bipartite, we can prove this statement by presenting an odd cycle in G.

1.2 Matchings

1.2.1 Matchings in Bipartite Graphs

A *matching* of *size* k in a graph G is a set of k pairwise disjoint edges. The vertices belonging to the edges of a matching are *saturated* by the matching; the others are *unsaturated*. If a matching saturates every vertex of G, then it is a *perfect matching*. A *maximum matching* is a matching of maximum size.

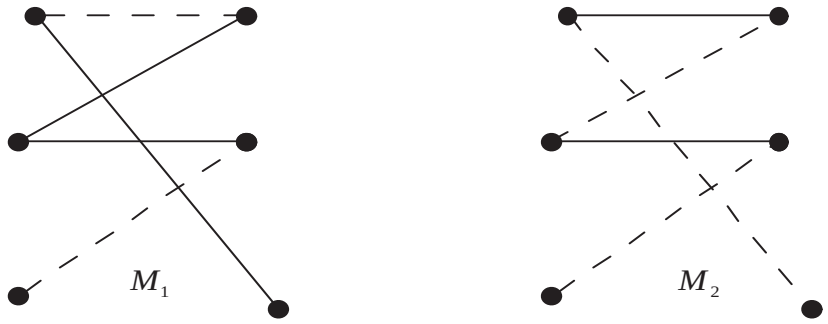

Figure 1.2: A matching M_1 and a perfect matching M_2.

In Figure 1.2 two matchings M_1 and M_2 in the same bipartite graph are

illustrated; the edges which belong to the matchings are indicated by dashed lines. In particular, M_2 is a perfect matching.

Given a matching M, an *M-alternating path* is a path that alternates between edges in M and edges not in M. An M-alternating path P that begins and ends at M-unsaturated vertices is an *M-augmenting path*; replacing $M \cap E(P)$ by $E(P) - M$ produces a new matching M' with one more edge that M. Maximum matchings are characterized by the absence of augmenting path, which was proven by in 1957.

Theorem 1.2 *(Berge [5])A matching M in a graph (not necessarily bipartite) G is a maximum matching in G if and only if G has no M-augmenting path.*

A *vertex cover* of G is a set S of vertices such that S contains at least one endpoint of every edge of G. The vertices in S "cover" the edges of G. The König-Egerváry Theorem states an equality relation between maximum matching and minimum vertex covering in bipartite graphs.

Theorem 1.3 *(König [20], Egerváry [10]) If G is a bipartite graph, then the maximum size of a matching in G equals the minimum size of a vertex cover of G.*

For any $S \subseteq X$, let $N(S)$ denote the set of vertices having a neighbor in S. If a matching M saturates X, it must be true that $|N(S)| \geq |S|$. This is known as Hall's condition. In 1935, the mathematician Philip Hall proved that this obvious necessary condition is also sufficient.

Theorem 1.4 *(Hall [16]) If G is a bipartite graph with bipartition X, Y, then G has a matching M that saturates X if and only if $|N(S)| \geq |S|$ for all $S \subseteq X$.*

When the sets of the bipartition have the same size, i.e., $|X| = |Y|$, then G has a perfect matching under Hall's condition.

Corollary 1.5 *If G is a bipartite graph with bipartition X, Y and $|X| = |Y|$, then G has a perfect matching if and only if $|N(S)| \geq |S|$ for all $S \subseteq X$.*

The *degree* of a vertex v in a graph G, written $d_G(v)$ or $d(v)$, is the number of non-loop edges containing v plus twice the number of loops containing v. The maximum degree is $\Delta(G)$; the minimum degree is $\delta(G)$. A graph G is *regular* if $\Delta(G) = \delta(G)$; *k-regular* if $\Delta(G) = \delta(G) = k$. The "first" Theorem of graph theory tells us the sum of vertex degrees is twice the number of edges. Thus every graph has an even number of vertices of odd degree.

If G is a k-regular bipartite graph, then it is easy to show that G satisfies Hall's condition, i.e. $|N(S)| \geq |S|$ for all $S \subseteq X$.

Corollary 1.6 *For $k > 0$, every k-regular bipartite graph has a perfect matching.*

1.2.2 Matchings in General Graphs

A graph G is *connected* if it has a path for each pair $u, v \in V(G)$; otherwise it is *disconnected*. A *subgraph* of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. An *induced subgraph* of G is a subgraph H of G such that $E(H)$ consists of all edges of G whose endpoints belong to $V(H)$. For any subset $S \subseteq V(G)$, $G - S$ denotes the induced subgraph on $V(G) - S$. The *components* of a graph G are its maximal connected subgraphs.

A *cut-edge* or *cut-vertex* of a graph is an edge or vertex whose deletion increases the number of components. The *order* of a graph G, written $n(G)$, is the number of vertices in G. An *odd component* of a graph is a component of odd order; the number of odd components of H is $o(H)$.

Now we shall expand our discussion of matchings to general graphs. Tutte

found a necessary and sufficient condition for an arbitrary graph to have a perfect matching.

Theorem 1.7 *(Tutte [30]) A graph G has a perfect matching if and only if $o(G - S) \leq |S|$ for every $S \subseteq V(G)$.*

Most applications of Tutte's Theorem involve showing that some other condition implies Tutte's condition, which then guarantees a perfect matching. Here is an example.

Corollary 1.8 *(Petersen [27]) Every 3-regular graph with no cut-edge has a perfect matching.*

In this thesis, we will be examining bipartite graphs and will not be using Tutte's Theorem.

1.3 Networks and Flows

A *network* is a digraph with a nonnegative capacity $c(e)$ on each edge e and a distinguished *source vertex* s and *sink vertex* t. Vertices are also called *nodes*. A *flow* f assigning a value $f(e)$ to each edge e. We write $f^+(v)$ or $f^+(S)$ for the total flow on edges exiting a node v or a set of nodes S; the total flow on entering edges is $f^-(v)$ or $f^-(S)$. In terms of $f(e)$, $f^+(v) = \sum_{e=(v,u)} f(e)$ and $f^-(v) = \sum_{e=(u,v)} f(e)$; $f^+(S) = \sum_{\substack{e=(v,u) \\ v \in S, u \notin S}} f(e)$ and $f^-(S) = \sum_{\substack{e=(u,v) \\ v \in S, u \notin S}} f(e)$.

A *feasible flow* satisfies the *capacity constraints* $0 \leq f(e) \leq c(e)$ for each edge and the *conservation constraints* $f^+(v) = f^-(v)$ for each node $v \notin \{s, t\}$. The *value val(f)* of a flow f is the net flow $f^-(t) - f^+(t)$ into the sink (or the net flow

$f^+(s) - f^-(s)$ out of the source). A *maximum flow* is a feasible flow of maximum value.

A *source/sink cut* partitions the nodes of a network into a *source set* S containing s and a *sink set* T containing t. The elements of the cut $[S,T]$ are the edges from S to T; every s,t*-path* uses at least one edge of $[S,T]$. The *capacity* of the cut $[S,T]$, written *cap(S,T)*, is the total of the capacities on the edges of $[S,T]$.

The most important result in network flows is the Max-flow Min-cut Theorem, proved by Ford and Fulkerson [12]. This result can be viewed as a special case of the strong duality property in Linear Programming.

Theorem 1.9 *(Max-flow Min-cut Theorem) In every network, the maximum value of a feasible flow equals the minimum capacity of a source/sink cut.*

Let us now apply the Max-flow Min-cut Theorem to get a quick proof of the König-Egerváry Theorem. Let G be a bipartite graph with bipartition X, Y. Build a network G' by directing all edges of G from X to Y, adding a new point s (the "source") joined to all points of X and a new point t (the "sink") to which all points of Y are joined. Assign capacity ∞ to all edges of G and capacity of 1 to all new edges of G'. It is easy to see that a maximum matching in G corresponds to a maximum flow in G', and so the maximum size of a matching equals the maximum value of a feasible flow. We need to show that the minimum size vertex cover in G equals the minimum capacity of a source/sink cut in G'. Assume $I \cup J$ is a minimum vertex cover in G, where $I \subseteq X$ and $J \subseteq Y$. Let $S = \{s\} \cup (X - I) \cup J$ and $T = \{t\} \cup I \cup (Y - J)$. Then $cap(S,T) = |I \cup J| = |I| + |J|$. We claim that $[S,T]$ is a minimum source/sink cut in G'. Otherwise, there exists some cut $[S',T']$ such that $cap(S',T') < cap(S,T)$. We may write $S' = \{s\} \cup (X - I') \cup J'$ and

$T' = \{t\} \cup I' \cup (Y - J')$ for some vertex cover $I' \cup J'$, where $I' \subseteq X$ and $J' \subseteq Y$. Then $|I'| + |J'| = cap(S', T') < cap(S, T) = |I| + |J|$. This leads to a contradiction.

In combinatorial applications, we typically have integer capacities and want a solution in which the flow on each edge is an integer.

Corollary 1.10 *(Integrality Theorem) If all capacities in a network are integers, then there is a maximum flow assigning integral flow to each edge.*

In Fulkerson, Hoffman and McAndrew [14], and Anstee [3], there is an exploration of how network flow techniques can assist in finding general matchings.

1.4 Capacitated Perfect b-matchings and Bidirected Flows

Let $G = (V, E)$ be a graph representing a communication network, i.e. the nodes v can be viewed as terminals and there is an edge $\{v, w\}$ if two terminals can directly communicate. Every node v has capacity/demand $b_v \in \mathbb{N}$ of possible information exchange operations. Now the task is to find an intersection of the terminals such that the demand is fulfilled.

We denote the number of uses of edge e by x_e and the set of edges having a node $v \in V$ as one of their endnodes by $\delta(v)$. A mapping $x : E \mapsto \mathbb{N}$ is called a b-matching if

$$x(\delta(v)) := \sum_{e \in \delta(v)} x_e \leq b_v \qquad \forall\ v \in V.$$

These conditions are called *node constraints* if $x(\delta(v)) = b_v$ for all v and a mapping fulfilling all these conditions is called a *perfect b-matching* on G. When $b_v = 1$ for all vertices in G, then (perfect) b-matchings are the usual (perfect) matchings. Now the problem is complicated by "capacity constraints" and "lower bounds" on the

edges, i.e. for every edge $e \in E$ there are two numbers l_e, u_e with $u_e \geq l_e \geq 0$, which give the maximal necessary usage and minimal possible usage of a single edge in the graph. Then a mapping x is called a *capacitated perfect b-matching* if it is a perfect b-matching on G and

$$l_e \leq x_e \leq u_e \qquad \forall\ e \in E.$$

Now the optimization problem occurs when a cost function $c : E \mapsto \mathbb{R}$ is introduced giving for every edge $e \in E$ the cost for exchanging one unit from the tail node of e to the head node of e. Then we want to find a capacitated perfect b-matching which minimizes total cost $c(x)$ where

$$c(x) := \sum_{e \in E} c_e x_e.$$

The above introduced *capacitated perfect b-matching problem* can be formulated as an integer linear programming problem.

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & x(\delta(v)) = b_v \ \forall\ v \in V \\ & l_e \leq x_e \leq u_e \ \forall\ e \in E \\ & x \in \mathbb{N}. \end{aligned}$$

A *bidirected graph* $G = (V, E)$ is a graph in which some edges (called *loops*) may have both ends incident with the same node, some edges (called *lobes*) may have only one end, and in which each edge has a direction associated with each node with which it is incident. Thus each edge has either one or two (not necessarily distinct) ends, and each of these ends is either a *tail end* or a *head end*.

For any vertex $v \in V$, we define for bidirected graphs

$\delta(v) :=$ the set of edges having exactly one end at v,

$\gamma(v) :=$ the set of edges having both ends at v,

$\delta^+(v) :=$ the set of edges in $\delta(v)$ whose single end at v is a head end,

$\delta^-(v) :=$ the set of edges in $\delta(v)$ whose single end at v is a tail end,

$\gamma^+(v) :=$ the set of loops having two head ends at v,

$\gamma^-(v) :=$ the set of loops having two tail ends at v.

Similarly, if we denote by $x_e$the number of units assigned to edge e, then we have to find a mapping $x : E \mapsto \mathbb{N}$ such that

$$x(\delta^+(v)) - x(\delta^-(v)) + 2x(\gamma^+(v)) - 2x(\gamma^-(v)) = b_v \ \forall \ v \in V.$$

These conditions are called *flow conservation constraints* and a mapping fulfilling all these conditions is called a *bidirected flow* on G. Now the problem may again be complicated by "capacity constraint" and "lower bounds" on the edges, i.e. for every edge $e \in E$ there are two numbers l_e, u_e with $u_e \geq l_e \geq 0$, which give the maximal necessary usage and minimal possible usage of a single edge in the network. Then a mapping x is called a *feasible bidirected flow* if it is a bidirected flow on G and

$$l_e \leq x_e \leq u_e \qquad \forall \ e \in E.$$

As in the b-matching problem we assume a cost function $c : E \mapsto \mathbb{R}$ giving for every edge e the cost for "transporting" one unit. Then the optimization problem is to find a feasible bidirected flow x which minimizes total cost $c(x)$ where

$$c(x) := \sum_{e \in E} c_e x_e.$$

The above introduced *bidirected flow problem* can also be formulated as an integer linear programming problem.

$$\min \quad \sum_{e \in E} c_e x_e$$

$$\text{s.t.} \quad x(\delta^+(v)) - x(\delta^-(v)) + 2x(\gamma^+(v)) - 2x(\gamma^-(v)) = b_v \ \forall \ v \in V$$

$$l_e \leq x_e \leq u_e \ \forall \ e \in E$$

$$x \in \mathbb{N}.$$

The bidirected flow problem looks quite similar to the capacitated b-matching problem. In fact a bidirected flow in a bidirected graph $G = (V, E)$ can be reduced to a capacitated b-matching in the undirected graph $G' = (V \cup V', \tilde{E})$ — with $V' = \{i' : i \in V\}$ and $\tilde{E} = \{\tilde{e} : e \in E\}$, where $\tilde{e}$ are given below — by doing the following (Derigs [6] and Edmonds [8]). Let $e = (i, j)$ be an edge in E.

- If e has 1 head for i and 1 head for j, then we construct one corresponding edge $\tilde{e} = (i', j')$. In particular, when $i = j$, i.e., e has 2 heads for i, then $\tilde{e} = (i', i')$, i.e., a loop at i'.

- If e has 1 tail for i and 1 tail for j, then we construct $\tilde{e} = (i, j)$. In particular, when $i = j$, i.e., e has 2 tails for i, then $\tilde{e} = (i, i)$, i.e., a loop at i.

- If e has 1 head for i and 1 tail for j, then we construct $\tilde{e} = (i', j)$. In particular, when $i = j$, i.e., e has 1 head and 1 tail for i, then $\tilde{e} = (i', i)$.

Thus a capacitated b-matching in a bidirected graph $G = (V, E)$ is an assignment of nonnegative integers x_e to the edges e of G such that x_e fulfills the "lower bound" and "capacity" conditions and the capacity/demand b_v at each node $v \in V$ is fulfilled, in the sense that edge e contributes $-2x_e, -x_e, 0, x_e, 2x_e$ units to the capacity/demand at v according to whether e has 2 tails, 1 tail, 0 ends, 1 head, 2 heads at v.

Chapter 2

Applications of Matchings

There are many applications of matchings in both bipartite and non-bipartite graphs, such as the Assignment Problem, the Chinese Postman Problem, etc.

2.1 Applications of Matchings in Bipartite Graphs

2.1.1 The Assignment Problem

Assignment of people or other resources to various workstations or certain equipment to accomplish certain tasks while minimizing the total cost is a frequent management decision problem. For example, assigning employees to various tasks, teachers to different classes, crews to projects, and ambulances to first-aid stations are good real-wold examples of assignment problems. The basic goal of the assignment problem is either to minimize the total cost of completing all the required tasks or to maximize the total payoff (or benefits) from the assignments. The following example is from Lee and Shim [22].

Example

Kabuki Electronics specializes in producing electronic scanners. The com-

pany received a special order from a large cash-register company for optical scanners. To produce important components required for the scanner, the company has selected four employees for assignment to four different machines to complete the necessary tasks. Because of different individual training and experience, the cost of successful completion of the given task to each machine is different for the four employees.

Only one employee can be assigned to each machine. Also, each task is completed independently by one employee on one machine. The assignment problem at Kabuki along with the cost involved for each employee to complete each task on each machine, is presented in Table 2.1. Notice that there are the same number of rows (employees) and columns (machines). Also, the amount of supply and demand for each row and each column is exactly one.

Employee \ Machine	A	B	C	D	Supply
1	20	10	14	13	1
2	18	10	22	18	1
3	28	14	20	28	1
4	6	10	22	8	1
Demand	1	1	1	1	4

Table 2.1: Tableau Form of the Kabuki Electronics Assignment Problem

In Table 2.1, we can easily see that it costs \$20 for employee 1 to complete the task using machine A, \$10 using machine B, and so on. The cost required for each employee to complete each task may be determined by the amount of time required and the amount of other resources (i.e., materials) used to complete the task. The objective of this assignment problem is to find the employee's optimum assignment schedule to each machine to minimize the total cost to complete the tasks.

The assignment problem is a weighted bipartite matching problem, where

the costs are the weights. In particular, the sets of the bipartition have the same size, i.e., there are equal numbers of employees and machines. Our Goal is to find the perfect matching between employees and machines such that the total cost is minimized. The best known method to solve assignment problems is the Hungarian Method. For details, please refer to Kuhn [21] or Munkres [25].

The optimal assignment and total cost of the Kabuki Electronics Assignment Problem are as follows:

Optimal Assignment		
Employee	Machine	Cost
1	D	13
2	B	10
3	C	20
4	A	6
		Total Cost = $49

Table 2.2: The Optimal Assignment of the Kabuki Electronics Assignment Problem

2.1.2 The Dimer Problem

The *close-packed dimer model* of statistical mechanics can be stated as follows. One considers a set of sites and a set of bonds connecting certain pairs of sites. Each bond b can absorb a "dimer" (which represents a diatomic molecule) with corresponding energy E_b. It is required that each site is occupied exactly once by one the atoms of a dimer. A state s is an arrangement of dimers which meets this requirement, and its energy $E(s)$ is $\sum E_b$ where the sum is taken over all bonds b which absorb a dimer. Then the partition function of the dimer model may be viewed as a density function of energy levels.

The dimer model was first considered by Roberts [29] in 1935, and by Fowler and Rushbrook [13]. The dimer model for 2-dimensional lattice graphs appears in

calculations of the thermodynamic properties of a system of diatomic molecules-dimers arranged in a planar sheet. The partition function can be evaluated in polynomial time using Kasteleyn's method [18] for enumerating perfect matchings in planar graphs.

One special type of graph for which enumeration of perfect matchings is of considerable "real-world" interest is the class of rectangular lattice graphs or chessboards (perhaps more accurately "go boards"). Consider an $m \times n$ chessboard with mn even. In how many ways can this board be covered by dominoes (dimers)?

In graph theoretic terms, the dimer problem for 2-dimensional lattice graphs can be formulated as follows. Let $L(m, n)$ denote the graph whose vertices are the squares (think of them as the centers of the squares) of an $m \times n$ chessboard with mn even, two being adjacent if and only if they have a boundary line in common. $L(m, n)$ is a bipartite graph, which means that its vertices may be partitioned into two sets Z_1, Z_2 such that if e is an edge of $L(m, n)$ then $|e \cap Z_1| = |e \cap Z_2| = 1$. Moreover, we have also that $|Z_1| = |Z_2| = mn/2$. Now, the problem is to determine the number $\Phi(L(m, n))$ of perfect matchings in $L(m, n)$. Kasteleyn [18] gave the answer:

$$\Phi(L(m,n)) = 2^{mn/2} \prod_{k=1}^{m} \prod_{l=1}^{n} \left(\cos^2(\frac{\pi k}{m+1}) + \cos^2(\frac{\pi l}{n+1}) \right)^{1/4}.$$

But the same problem for 3-dimensional lattices remains an important open problem of statistical physics (see Kasteleyn [19] for references).

2.2 Applications of Matchings in General Graphs

2.2.1 The Chinese Postman Problem

A postman has to leave a post office and traverse a number of streets at least once for delivering and picking up mails, then return to the post office to do his job. How should he plan his walk to minimize his walking distance or time? We can phrase it as a graph problem as follows: A postman traverses all edges in a road network, starting and ending at the same vertex. The edges have nonnegative weights representing distance or time. We seek a closed walk of minimum total length that covers each edge at least once. This is called the *Chinese Postman Problem* in honor of the Chinese mathematician Meigu Guan [15], who proposed it.

Recall that a *closed trail* is a trail whose length is at least one and its endpoints are equal. We say that a graph whose edges comprise a single closed trail is *Eulerian*. We use the term *circuit* as another name for "closed trail"; a circuit containing every edge of G is an *Eulerian circuit*. We also use *odd vertex* or *even vertex* to indicate the parity of a vertex degree.

If every vertex in the road network is even, then we can show that the graph G is Eulerian and so the minimum total length is the sum of the weights. Otherwise, we must repeat edges. If there are only two odd vertices, then we can use Dijkstra's Algorithm [7] to find the shortest path between them and solve the problem. If there are $2k$ ($k \geq 2$) odd vertices, then we can use the shortest path algorithm to find the shortest paths connecting each pair of odd vertices. We can use these lengths as weights on the edges of K_{2k}, and then our problem is to find the minimum total weight of k edges that pair up these $2k$ vertices. This is a weighted version of the maximum matching problem. Edmonds and Johnson [9] described a way to solve

this Problem.

An interesting application of the Chinese Postman Problem is the Plotter Problem which was introduced by Asano, Edahiro, Imai, Iri and Murota [4]. The problem is the plotting of a large figure on a computer controlled plotter. They draw straight line segments between points to draw a piecewise linear curve then lift the pen up and travel to the next curve to be drawn. We may think of the plotter as a postman with a pen. In the Chinese Postman Problem we form an auxiliary complete graph defined on the vertices of odd degree. Each edge has a weight, namely the distance between the vertices in the original graph. In the Plotter Problem this weight is the L_∞ distance, $d((x_1, y_1), (x_2, y_2)) = \max\{|x_1 - x_2| + |y_1 - y_2|\}$. It is worth pointing out that the graph G in the Chinese Postman Problem must be connected (since the postman can only walk) while the graph G' in the Plotter Problem may be disconnected. If G' is connected, the solution is then the same as the that for the Chinese Postman Problem. Otherwise, we need first to add some extra edges carefully to make it connected. In fact, a minimal number of edges required to join components will correspond to the Travelling Salesman Problem (TSP).

2.2.2 Quadrilateral Mesh Generation in Computer-Aided Design

Müller-Hannemann [26] considered the following definition of polygon for use in mesh generation. A *polygon* is a region in the plane or, more generally, of a smooth surface in the 3-dimensional space, bounded by a finite, closed sequence of straight line segments or curves (*edges*). A polygon is *simple* if its edges do not cross each other, and *convex* if the internal angle at each vertex is at most π. A *mesh* is a set of openly disjoint, convex and simple polygons.

Müller-Hannemann [26] has considered a mesh refinement problem arising in the computer-aided design (CAD) of motor vehicle parts, for example, bodyworks or chassis or their components: a coarse mesh of curved polygons, which approximates the surface of a workpiece, has to be refined into quadrilaterals such that the resulting mesh can be used for a numerical analysis.

Originating from a concrete application, [26] has described the difficult modeling process, analyze the computational complexity of the mesh refinement problem in various forms, and develop a new algorithmic approach. In [26], we see an original application of network flow and matching techniques in a mesh refinement problem. The experiments in [26] showed that this approach usually leads to meshes with a very good quality.

The mesh generation problem can be reduced to a bidirected flow problem on a bidirected graph G with capacity constraints and lower bounds on the edges and some additional node balance conditions (Müller-Hannemann [26]). We have shown in section 1.4 that the bidirected flow problem can be reduced to the capacitated perfect b-matching problem. There are several polynomial algorithms for b-matching problems (see Pulleyblank [28], Anstee [2], and Miller and Pekny [24]).

Chapter 3

Introduction of the Problem

3.1 Robustness

Network robustness measures how well a network can continue to perform its function after node or arc failures. There are several different ways and methodologies to measure robustness of networks, e.g., resilience to random node failures (fragility), resilience to attacks (vulnerability), or number of alternative paths.

The idea of "robustness" considered here is very different from network robustness. We study the robustness of a graph with respect to the property of having a perfect matching under the operations of vertex deletion.

This thesis studies lattice graphs which are readily seen to have many perfect matchings and considers whether if we delete vertices the resulting graphs continue to have perfect matchings. In order to guarantee the resulting graphs still have perfect matchings, we require some constraints on the deleted vertices. We will impose the constraint that the deleted vertices are at a certain distance from each other.

3.2 Motivation

This thesis was motivated by a result of Jamison and Lochner [17]. They studied a problem about tiling fringed chessboards with dominoes. A *fringe* on a $p \times q$ chessboard is a set of additional nonadjacent squares added above the top row. The squares may be considered as colored black and white consistently with the board. If the fringed board is to be tiled by dominoes, then the total number of black squares must equal the total number of white squares. In general, this parity condition is not sufficient. However, if the board is deep enough, i.e., p is big enough, then this parity condition does suffice to guarantee a tiling.

A *balanced fringe* is a fringe that contains an equal number of black and white squares. The *depth* of a fringe X is defined as $\Delta(X) = \max_{1 \le i \le q} |B_i - W_i|$, where B_i is the number of black squares up to and including i and W_i is the number of white squares up to and including i. We state the main result of [17] without proof.

Theorem 3.1 *Let B be a fringed board with a balanced fringe X obtained from a $p \times q$ chessboard by adjoining X above the top row. If $p = 2\Delta(X)$, then B has a domino tiling.*

In order to tile a fringed board by dominoes, we first need to pair the squares in the fringe with their adjacent squares in the $p \times q$ chessboard. Now, we delete these pairs (i.e., the fringe and its adjacent squares) from the fringed board. Note that to cover a fringed board by dominoes is equivalent to cover the resulting board after deletion. From the point of view of tiling with dominoes, adding some nonadjacent squares (i.e., a fringe) above the top row to a $p \times q$ chessboard is equivalent to deleting their adjacent squares from the top row of the chessboard.

In graph theoretic terms, given a $p \times q$ (pq even) lattice graph, we delete

an equal number of black and white vertices from the top row. Theorem 3.1 says that if p is big enough then the resulting graph still has a perfect matching. In particular, $q = p$ (p even) will work. Now, we consider a more general version of vertex deletion. A $p \times p$ lattice graph (p even) is easily seen to have many perfect matchings. Consider deleting an equal number of black and white vertices in this lattice graph (not restricted to the top row). We want the resulting graph continues to have a perfect matching. In order to avoid local problems (e.g. deleting all neighbors of a given vertex will result in a graph that has no perfect matching), we require that the deleted vertices to be mutually far apart. We will be more specific in Lemma 4.3.

3.3 Definitions and Notations

The *n-dimensional lattice graph* $Q(m, n)$ is the following graph: $Q(m, n)$ has vertices $(x_1, x_2, \ldots, x_n)$, where $x_i \in \{1, 2, \ldots, m\}$ for $i = 1, 2, \ldots, m$, and edges $\big((x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n), (x_1, \ldots, x_{i-1}, x_i+1, x_{i+1}, \ldots, x_n)\big)$, where $x_j \in \{1, 2, \ldots, m\}$ for $j = 1, 2, \ldots, n$ and $x_i \neq m$.

For example, $Q(m, n = 2)$ gives an $m \times m$ lattice graph divided into m^2 vertices of two alternating colors — black and white. Figure 3.1 illustrates a 2-dimensional lattice graph with $m = 20$.

To be precise, let's say the vertex (1,1,...,1) is a white vertex. Then $(x_1, x_2, \ldots, x_n)$ is a white vertex if and only if

$$\sum_{i=1}^{n} x_i \equiv n \pmod 2.$$

Then $Q(m, n)$ is a bipartite graph with bipartition W and B, where W is the set of all white vertices and B is the set of all black vertices.

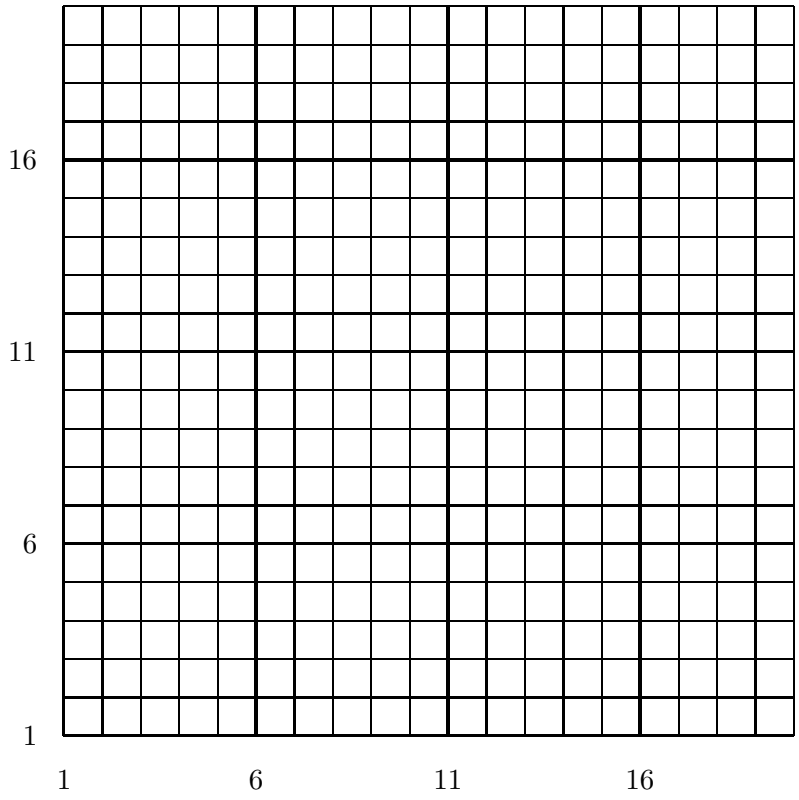

Figure 3.1: $Q(m, 2)$ gives an $m \times m$ lattice graph, here $m = 20$.

If m is odd, then the total number of vertices in $Q(m, n)$ is m^n, which is also odd and so there is no perfect matching for $Q(m, n)$. In order to have a perfect matching in $Q(m, n)$, we may assume m is even. Then $|W| = |B| = \dfrac{|Q(m, n)|}{2} = \dfrac{m^n}{2}$.

For any $S \subseteq W$, the *neighbor set* of S in $Q(m, n)$, denoted by $N(S)$, is the set of all vertices adjacent to the vertices in S. Since all the vertices adjacent to S are black, then $N(S) \subseteq B$. The *surplus* of S is $|N(S)| - |S|$.

Given $S \cup N(S)$ is connected, the *side lengths* of $S \cup N(S)$ are

$$a_i = \max\{x_i : (x_1, x_2, \ldots, x_n) \in S \cup N(S)\} - \min\{x_i : (x_1, x_2, \ldots, x_n) \in S \cup N(S)\},$$

where $i = 1, 2, \ldots, n$. We define the distance of two vertices $(x_1, x_2, \ldots, x_n)$ and

$(x'_1, x'_2, \ldots, x'_n)$ as

$$d\big((x_1, x_2, \ldots, x_n), (x'_1, x'_2, \ldots, x'_n)\big) = \sum_{i=1}^{n} |x_i - x'_i|.$$

Choose $k \in \{1, 2, \ldots, m\}$. Then fix $x_i = k$ and define a *complete plane* by

$$\pi_{\{x_i=k\}} = \{(x_1, x_2, \ldots, x_{i-1}, k, x_{i+1}, \ldots, x_n) : x_j = 1, 2, \ldots, m \text{ for } j \neq i\}.$$

A complete plane $\pi_{\{x_i=k\}}$ contains all the points where $x_i = k$. Otherwise, we say a plane is *incomplete*. In particular, when $n = 2$, fix i, the set

$$\pi_{\{x_1=i\}} = \{(i, x_2) : x_2 = 1, 2, \ldots, m\}$$

gives a *complete column* C_i (we use the word column as it is more appropriate in 2 dimensions). Similarly, we can define a complete row. We define the *surplus of column* C (not necessarily complete) as $|C \cap N(S)| - |C \cap S|$.

Let B' be the set of deleted black vertices in B, and W' be the set of deleted white vertices in W.

3.4 A Result in the 2-D Case

The following Theorem is a result of Aldred, Anstee and Locke [1].

Theorem 3.2 *Assume m is an even integer. Let $Q(m, 2)$ be a 2-dimensional lattice graph with bipartition B, W. Choose any sets $B' \subseteq B, W' \subseteq W$. If B', W' have the following three properties:*

(a) $|B'| = |W'|$,

(b) $d\big((x_1, x_2), (x'_1, x'_2)\big) > d$, *for* $(x_1, x_2), (x'_1, x'_2) \in B'$,

(c) $d\big((x_1, x_2), (x'_1, x'_2)\big) > d$, *for* $(x_1, x_2), (x'_1, x'_2) \in W'$,

where $d = 4\lceil\sqrt{m}\,\rceil + 8$*, then the graph* $Q(m,2) - (B' \cup W')$ *has a perfect matching.*

■

Theorem 3.2 gives a result for the 2-dimensional case. We try to get some similar results in higher dimensions in next chapter.

Chapter 4

Main Result for n-dimensional Lattice Graphs

4.1 Statement of Theorem 4.1

Theorem 4.1 is a nontrivial analogue of Theorem 3.2 for higher dimensions. We will use some proof ideas from the proof of Themreom 3.2 [1], but the overall proof of Theorem 4.1 is quite different from that of Theorem 3.2.

In this whole chapter, we assume m, n are natural numbers and m is even, $n \geq 2$.

Theorem 4.1 *Let $Q(m,n)$ be an n-dimensional lattice graph with bipartition B, W. Choose any sets $B' \subseteq B, W' \subseteq W$. If B', W' have the following three properties:*

(a) $|B'| = |W'|$,

(b) $d\big((x_1, x_2, \ldots, x_n), (x'_1, x'_2, \ldots, x'_n)\big) > d_n$, $\forall\ (x_1, x_2, \ldots, x_n),\ (x'_1, x'_2, \ldots, x'_n) \in B'$,

(c) $d\big((x_1, x_2, \ldots, x_n), (x'_1, x'_2, \ldots, x'_n)\big) > d_n$, $\forall\ (x_1, x_2, \ldots, x_n),\ (x'_1, x'_2, \ldots, x'_n) \in W'$,

where $d_n = 4n(n+1)\lceil\sqrt{m}\,\rceil$, *then the graph* $Q(m,n) - (B' \cup W')$ *has a perfect matching.* ■

Before proving the theorem, we will prove 4 lemmas. The proofs of Lemma 4.2 and 4.3 are loosely based on arguments in [1]. The results of Lemma 4.4 and 4.5 are original, and the proofs are quite different from those of Lemma 4.2 and 4.3.

Recall that if $S \cup N(S)$ is connected, the *side lengths* of $S \cup N(S)$ are

$$a_i = \max\{x_i : (x_1, x_2, \ldots, x_n) \in S \cup N(S)\} - \min\{x_i : (x_1, x_2, \ldots, x_n) \in S \cup N(S)\},$$

where $i = 1, 2, \ldots, n$. Without loss of generality, we may assume $a_1 \geq a_2 \geq \cdots \geq a_n$. We first expand the region $S \cup N(S)$ to an $a_1 \times a_2 \times \cdots \times a_n$ grid. Then we expand the $a_1 \times a_2 \times \cdots \times a_n$ grid to an $(a_1 + d_n) \times (a_2 + d_n) \times \cdots \times (a_n + d_n)$ grid by adding $d_n/2$ at each of $2n$ directions, where d_n is even. For any x in the $a_1 \times a_2 \times \cdots \times a_n$ grid, we define

$$B_n(x, d) = \{y \in Q(m,n) : d(x,y) \leq d\}.$$

Then $B_n(x, d_n/2)$ is contained in the $(a_1 + d_n) \times (a_2 + d_n) \times \cdots \times (a_n + d_n)$ grid since any point of $B_n(x, d_n/2)$ is at most $d_n/2$ from x and hence at most $d_n/2$ from the $a_1 \times a_2 \times \cdots \times a_n$ grid.

Lemma 4.2 *For any vertex* x *in* $Q(m,n)$, *we have*

$$\Big|B_n(x, d_n/2)\Big| \geq 2\Big(\frac{d_n}{2n}\Big)^n.$$

Proof: We prove it by induction. When $n = 2$, $B_2(x, d_2/2)$ gives a 2-dimensional "diamond". See Figure 4.1, the vertices which belong to the diamond are indicated by thick dots. The number of vertices in such a diamond is

$$\Big|B_2(x, d_2/2)\Big| = {d_2}^2/2 + d_2 + 1 \geq {d_2}^2/8 = 2\Big(\frac{d_2}{4}\Big)^2.$$

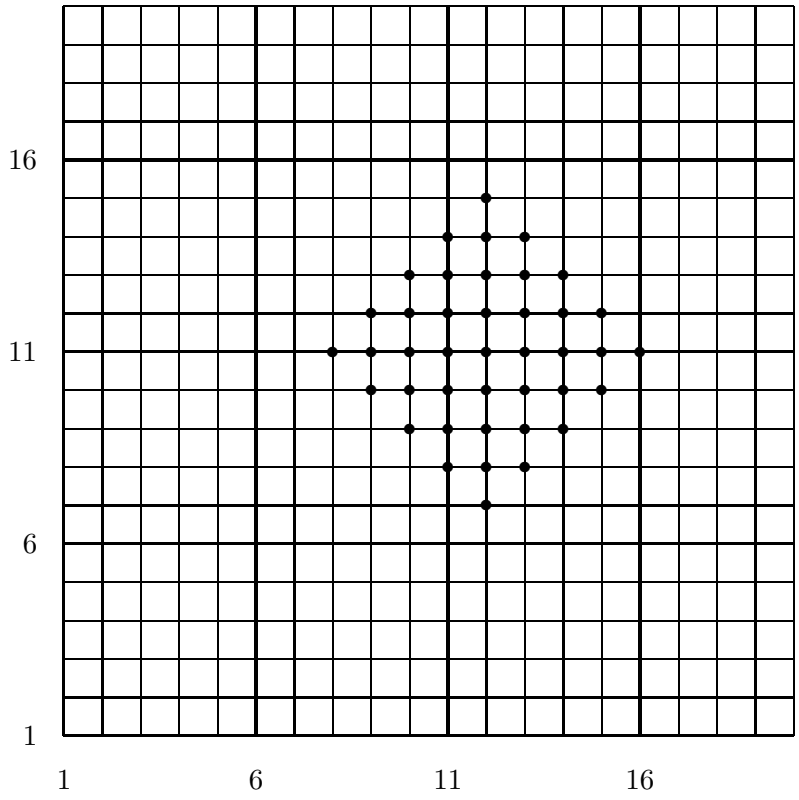

Figure 4.1: A 2-D diamond with $d = 8$ in the checkerboard with $m = 20$.

When $n = 3$, $B_3(x, d_3/2)$ gives a 3-dimensional "diamond". We may consider the origin point as the center and d as the diameter of the diamond in 3 dimensions. See Figure 4.2 for a sketch of a 3-dimensional diamond. We only count vertices in the upper half of the diamond. So,

$$\Big|B_3(x, d_3/2)\Big| \geq \frac{1}{3}\Big|B_2(x, d_3/2)\Big|(d_3/2+1) \geq \frac{1}{3}2\Big(\frac{d_3}{4}\Big)^2(d_3/2+1) \geq 2\Big(\frac{d_3}{6}\Big)^3.$$

For general n, we have

$$\begin{aligned}
\Big|B_n(x, d_n/2)\Big| &\geq \frac{1}{n}\Big|B_{n-1}(x, d_n/2)\Big|(d_n/2+1) && \text{only count the upper half} \\
&\geq \frac{1}{n}2\left(\frac{d_n}{2(n-1)}\right)^{n-1}(d_n/2+1) && \text{by induction} \\
&\geq 2\Big(\frac{d_n}{2n}\Big)^n. \qquad \blacksquare
\end{aligned}$$

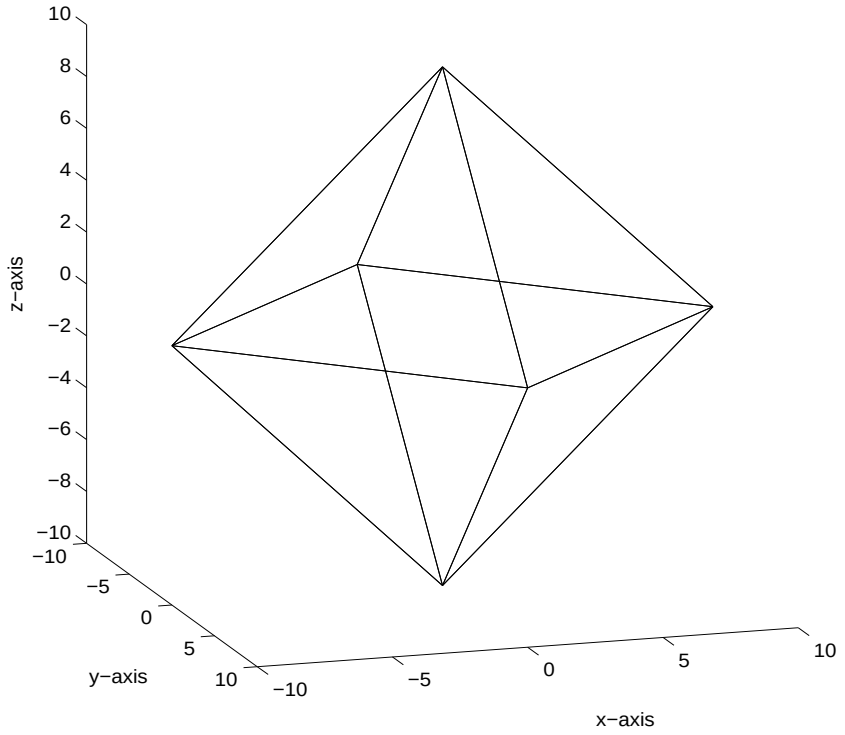

Figure 4.2: A sketch of a 3-D diamond.

The result $|B_n(x, d_n/2)| \geq 2\left(\frac{d_n}{2n}\right)^n$ is enough to prove Theorem 4.1, where $d_n = 4n(n+1)\lceil\sqrt{m}\,\rceil$. Actually, we can count $|B_n(x, d_n/2)|$ exactly for small n, e.g.,

$$\left|B_2(x, \frac{d_2}{2})\right| = \frac{{d_2}^2}{2} + d_2 + 1 \geq \frac{{d_2}^2}{2!}$$

and

$$\left|B_3(x, \frac{d_3}{2})\right| = \frac{1}{6}{d_3}^3 + \frac{1}{2}{d_3}^2 + \frac{4}{3}d_3 + 1 \geq \frac{{d_3}^3}{3!}.$$

It is easy to see that

$$\left|B_n(x, \frac{d_n}{2})\right| \quad = \quad \left|B_{n-1}(x, \frac{d_n}{2})\right| + 2 \sum_{\substack{i=0 \\ i \text{ is even}}}^{d_n-2} \left|B_{n-1}(x, \frac{i}{2})\right|$$

Thus approximately, $|B_n(x, d_n/2)| \geq \frac{{d_n}^n}{n!}$. However, even if we could show $|B_n(x, d_n/2)| \geq \frac{{d_n}^n}{n!}$, this would only improve d_n by a constant factor in Theorem 4.1.

4.2 Proof of Theorem 4.1

We first consider a special case in 2 dimensions when $|S \cup N(S)| \leq \frac{3}{4}m^2$. The main proof idea of Lemma 4.3 is from the proof of Theorem 3.2 [1].

Lemma 4.3 *Let $Q(m,2)$ be a 2-dimensional lattice graph with bipartition B, W. Choose any sets $S \subseteq W$, $B' \subseteq B$. If B', W' have the following two properties:*

(a) $|S \cup N(S)| \leq \frac{3}{4}m^2$,

(b) $d\big((x_1,x_2),(x_1',x_2')\big) > d_2 = 24\lceil\sqrt{m}\,\rceil$, *for* $(x_1,x_2),(x_1',x_2') \in B'$,

then

$$|B' \cap N(S)| \leq |N(S)| - |S|. \tag{4.1}$$

Proof: First we consider the case when $S \cup N(S)$ is connected. Then we can construct the $(a_1 + d_2) \times (a_2 + d_2)$ grid as described earlier.

If $a_2 < m$, then there is no complete column. Note that the surplus of every column is at least 0 since in any column the number of black vertices is $\geq$ the number of white vertices. Every two adjacent incomplete nonempty columns have a surplus of at least 1, since in only these two columns, the number of black vertices is more than the number of white vertices at least by 1. Now we pair all the columns from left to right. If a_1 is even, we have totally $a_1/2$ pairs and so the surplus is at least $a_1/2$; if a_1 is odd, we have $(a_1 - 1)/2$ pairs (1 column left) and so the surplus is at least $(a_1 - 1)/2$. Hence, the surplus of the a_1 incomplete nonempty columns is at least $\lfloor a_1/2 \rfloor$.

If $a_2 = m$, then $a_1 = a_2 = m$. Since $|S \cup N(S)| \leq \frac{3}{4}m^2$, $S \cup N(S)$ has at most $\lfloor \frac{3}{4}m^2/m \rfloor = \lfloor 3m/4 \rfloor$ complete columns. Hence, there are at least $m - \lfloor 3m/4 \rfloor = \lceil m/4 \rceil$ incomplete nonempty columns. Again, pair all the columns from left to

right and we obtain $a_1/2 = m/2$ pairs. There are 3 different types of pairs: a pair of two complete columns, a pair of one complete column with one incomplete nonempty columns and a pair of two incomplete nonempty columns. The first type pair has a surplus at least 0; the second type has a surplus at least 1 and the third type has a surplus at least 1. Since we have at least $\lceil m/4 \rceil$ incomplete nonempty columns, then the surplus is at least $\lfloor \frac{\lceil m/4 \rceil}{2} \rfloor \geq \lfloor \frac{m/4}{2} \rfloor = \lfloor m/8 \rfloor = \lfloor a_1/8 \rfloor$. Therefore, $|N(S)| - |S| \geq \lfloor a_1/8 \rfloor$.

We want to show $|B' \cap N(S)| \leq \lfloor a_1/8 \rfloor$. If $a_1 < d_2/2$ then at most one black vertex can be deleted. So we only need to consider the cases when $a_1 \geq d_2/2$, i.e. $d_2 \leq 2a_1$. Note that for $x, y \in B'$, $B(x, d_2/2) \cap B(y, d_2/2) = \emptyset$. Thus the number of deleted black vertices

$$\begin{aligned} |B' \cap N(S)| &\leq \left\lfloor \frac{(a_1 + d_2)(a_2 + d_2)}{B_2(x, d_2/2)} \right\rfloor \\ &\leq \left\lfloor \frac{(a_1 + d_2)^2}{{d_2}^2/8} \right\rfloor \\ &\leq \left\lfloor \frac{(3a_1)^2}{{d_2}^2/8} \right\rfloor = \left\lfloor \frac{72{a_1}^2}{{d_2}^2} \right\rfloor \end{aligned}$$

Now $d_2 = 24\lceil \sqrt{m} \rceil$, so ${d_2}^2 \geq 576m \geq 576a_1$ and so $\frac{72{a_1}^2}{{d_2}^2} \leq a_1/8$. Thus, $|B' \cap N(S)| \leq |N(S)| - |S|$.

Second we consider the case that $S \cup N(S)$ is disconnected, we may write $S \cup N(S) = \sum_{i=1}^{n} (S_i \cup N(S_i))$, where every $S_i \cup N(S_i)$ is connected and $(S_i \cup N(S_i)) \cap (S_j \cup N(S_j)) = \emptyset$ for $i \neq j$. By the above proof, we get $|B' \cap N(S)| = \sum_{i=1}^{n} |B' \cap N(S_i)| \leq \sum_{i=1}^{n} (|N(S_i)| - |S_i|) = \sum_{i=1}^{n} |N(S_i)| - \sum_{i=1}^{n} |S_i| = |N(S)| - |S|$. ■

We use the complete plane

$$\pi_{\{x_n = i\}} = \{(x_1, x_2, \ldots, x_{n-1}, i) : x_j = 1, 2, \ldots, m \text{ for } j \neq n\}$$

to cut $S \cup N(S)$. Denote the intersection by *layer* L_i:

$$L_i = (S \cup N(S)) \bigcap \pi_{\{x_n = i\}}.$$

Lemma 4.4 *Let $Q(m,n)$ be an n-dimensional lattice graph with bipartition B, W. Choose any set $S \subseteq W$. If S has the following two properties:*

(a) $|S \cup N(S)| \leq \frac{2^{n-1}+1}{2^n} m^n$,

(b) there exists some layer L_i such that $|L_i| > \frac{2^{n-2}+1}{2^{n-1}} m^{n-1}$,

then

$$|N(S)| - |S| \geq \left\lceil \frac{1}{2^{n+1}} m^{n-1} \right\rceil.$$

Proof: Let $a_1, a_2, \ldots, a_n$ be the side lengths of $S \cup N(S)$. We may assume $a_1 \geq a_2 \geq \ldots \geq a_n$. If $a_n < m$, then we can show that surplus $|N(S)| - |S| \geq \lfloor \frac{|L_i|}{2} \rfloor = \lfloor \frac{2^{n-2}+1}{2^n} m^{n-1} \rfloor$ by the fact that the number of black vertices $\geq$ the number of white vertices in any layer and the following "projection line" idea. Since $a_n < m$, then at least one of the two layers L_1 and L_m is empty. Assume $L_m = \emptyset$. Fix $x_1^0, x_2^0, \ldots, x_{n-1}^0$, the vertex set

$$\{(x_1^0, x_2^0, \ldots, x_{n-1}^0, j), j = i, i+1, \ldots, m\} \cap (S \cup N(S))$$

defines a *projection line*. Any projection line that starts with a black vertex has surplus ≥ 1; any projection line that starts with a white vertex has surplus ≥ 0. There are at least $\lceil \frac{|L_i|}{2} \rceil = \lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \rceil$ black vertices in L_i and so the projection lines which start from L_i give surplus $\geq \lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \rceil$. Note that the number of black vertices $\geq$ the number of white vertices in any other layers below L_i , so $|N(S)| - |S| \geq \lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \rceil$.

If $a_n = m$, then $a_1 = a_2 = \ldots = a_n = m$. Since $|S \cup N(S)| \leq \frac{2^{n-1}+1}{2^n} m^n$, then there exists some layer L_j such that $|L_j| \leq \lfloor \frac{2^{n-1}+1}{2^n} m^{n-1} \rfloor$. Then we can show that $|N(S)| - |S| \geq \lceil \frac{1}{2^{n+1}} m^{n-1} \rceil$ by the following "projection segment" idea. We may assume $i < j$. Fix $x_1^0, x_2^0, \ldots, x_{n-1}^0$, the vertex set

$$\{(x_1^0, x_2^0, \ldots, x_{n-1}^0, k), i \leq k \leq j\} \cap (S \cup N(S))$$

defines a *projection segment* between layer L_i and L_j.

Now we first count the surplus between layer L_i and L_j, inclusive. We have two cases which depend on the parity of $i + j$.

Case 1: $i + j \equiv 0 \pmod 2$.

1. Every projection segment which starts with a black vertex in L_i (or L_j) has surplus ≥ 1.

2. Every projection segment which starts with a white vertex in L_i (or L_j) and ends with a white vertex in L_j (or L_i) has surplus ≥ -1.

3. Every projection segment which starts with a white vertex in L_i (or L_j) and doesn't reach L_j (or L_i) has surplus ≥ 0.

There are at least $\lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \rceil$ black vertices in L_i and at most $\lfloor \frac{2^{n-1}+1}{2^{n+1}} m^{n-1} \rfloor$ white vertices in L_j . So we have surplus at least

$$1 \times \left\lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \right\rceil + (-1) \times \left\lfloor \frac{2^{n-1}+1}{2^{n+1}} m^{n-1} \right\rfloor = \left\lceil \frac{1}{2^{n+1}} m^{n-1} \right\rceil .$$

Case 2: $i + j \equiv 1 \pmod 2$.

1. Every projection segment which starts with a black vertex in L_i (or L_j) and ends with a white vertex in L_j (or L_i) has surplus ≥ 0.

2. Every projection segment which starts with a white vertex in L_i (or L_j) has surplus ≥ 0.

3. Every projection segment which starts with a black vertex in L_i (or L_j) and doesn't reach L_j (or L_i) has surplus ≥ 1.

So the surplus is

$$1 \times \left(\left\lceil \frac{2^{n-2}+1}{2^n} m^{n-1} \right\rceil - \left\lfloor \frac{2^{n-1}+1}{2^{n+1}} m^{n-1} \right\rfloor \right) = \left\lceil \frac{1}{2^{n+1}} m^{n-1} \right\rceil .$$

Therefore, the surplus between layers L_i and L_j inclusive is at least $\lceil \frac{1}{2^{n+1}} m^{n-1} \rceil$. Since the number of black vertices $\geq$ the number of white vertices in any other layers above L_j or below L_i , we have at least $\lceil \frac{1}{2^{n+1}} m^{n-1} \rceil$ surplus. ■

Lemma 4.5 *Let $Q(m,n)$ be an n-dimensional lattice graph with bipartition B, W. Choose any set $S \subseteq W$. If S satisfies $|S \cup N(S)| \leq \frac{2^{n-1}+1}{2^n} m^n$, then (4.1) holds.*

Proof: Note that $d_n = 4n(n+1)\lceil \sqrt{m}\, \rceil > d_{n-1} = 4(n-1)n\lceil \sqrt{m}\, \rceil$. We prove it by induction. Lemma 4.3 showed the case when $n = 2$. In the n-dimension, if $|L_i| \leq \frac{2^{n-2}+1}{2^{n-1}} m^{n-1}$ for all $i = 1, 2, \ldots, m$, then we are done by induction. Otherwise, there exists some L_i such that $|L_i| > \frac{2^{n-2}+1}{2^{n-1}} m^{n-1}$. By Lemma 4.4, we have $|N(S)| - |S| \geq \lceil \frac{1}{2^{n+1}} m^{n-1} \rceil$.

We want to show $|B' \cap N(S)| \leq \lceil \frac{1}{2^{n+1}} m^{n-1} \rceil$. If $m < d_n/n$ then at most one black vertex can be deleted. So we only need to consider the cases when $m \geq d_n/n$, i.e. $d_n \leq nm$. Note that for $x, y \in B'$, $B(x, d_n/2) \cap B(y, d_n/2) = \emptyset$. Thus the

number of deleted black vertices

$$\begin{aligned} |B' \cap N(S)| &\leq \left\lfloor \frac{(a_1+d_n)(a_2+d_n)\cdots(a_n+d_n)}{B_n(x,d_n/2)} \right\rfloor \\ &\leq \left\lfloor \frac{(m+d_n)^n}{2(d_n/2n)^n} \right\rfloor \\ &\leq \left\lfloor \frac{(m+nm)^n}{2(d_n/2n)^n} \right\rfloor \\ &= \left\lfloor \frac{(2n(n+1))^n m^n}{2{d_n}^n} \right\rfloor \end{aligned}$$

Now $d_n = 4n(n+1)\lceil\sqrt{m}\,\rceil \geq 4n(n+1)\lceil m^{1/n}\,\rceil \geq 4n(n+1)m^{1/n}$, so ${d_n}^n \geq (4n(n+1))^n m$ and so

$$\frac{(2n(n+1))^n m^n}{2{d_n}^n} \leq \frac{(2n(n+1))^n m^n}{24n(n+1)^n m} \leq \frac{1}{2^{n+1}} m^{n-1}.$$

Thus, $|B' \cap N(S)| \leq |N(S)| - |S|$. ■

From the proof of Lemma 4.5, we see that if there is some layer L_i such that $|L_i| > \frac{2^{n-2}+1}{2^{n-1}} m^{n-1}$, then $d_n = 4n(n+1)\lceil m^{1/n}\,\rceil$ is sufficient to guarantee (4.1). However, when every layer is small ($\leq \frac{2^{n-2}+1}{2^{n-1}} m^{n-1}$), we rely on induction. And we start the induction with $n = 2$, so we require $d_n = 4n(n+1)\lceil\sqrt{m}\,\rceil$ for every n. Although we don't know how to prove it, we believe that the best possible $d_n = f(n)\lceil m^{1/n}\,\rceil$, where $f(n)$ is a function of n. Here is an example to show that why the bound $m^{1/n}$ is best possible.

We choose $S = E \cap W$, where

$$E = \Big\{(x_1, x_2, \cdots, x_n) \mid x_n \in \{1, 2, \cdots, \frac{m}{2}\},\ x_i \in \{1, 2, \cdots, m\} \text{ for } i \neq n\Big\}.$$

Since the number of white vertices = the number of black vertices in E, the only surplus are those black vertices above E, then $|N(S)| - |S| = \frac{m^{n-1}}{2}$. Every deleted black vertex takes up a space of size at most ${d_n}^n$, then the maximum number of

black vertices which can be deleted is

$$|B' \cap N(S)| \geq \frac{m^n}{2}/{d_n}^n = \frac{m^n}{2{d_n}^n}.$$

Remark that we can choose W' in the other half. In order to make the resulting graph (after deletion) still have a perfect matching, it is necessary that

$$|B' \cap N(S)| \leq |N(S)| - |S|, \text{ i.e., } \frac{m^n}{2{d_n}^n} \leq \frac{m^{n-1}}{2} \Rightarrow d_n \geq m^{1/n}$$

Next, we give the proof of Theorem 4.1.

Proof of Theorem 4.1: We prove Theorem 4.1 by using Hall's Theorem on $Q(m,n) - (B' \cup W')$. Choose any subset $S \subseteq W$. We need to show that

$$|N(S) - B'| \geq |S|, \tag{4.2}$$

which is equivalent to (4.1).

Lemma 4.5 proves (4.1) for $|S \cup N(S)| \leq \frac{2^{n-1}+1}{2^n}m^n$. It is easy to see (4.1) holds for $|S \cup N(S)| \leq \frac{1}{2}m^n$ since $\frac{1}{2}m^n < \frac{2^{n-1}+1}{2^n}m^n$.

Next, we show (4.1) for $|S \cup N(S)| > \frac{1}{2}m^n$. Let $T = B - N(S)$, then $|T \cup N(T)| < \frac{1}{2}m^n$. Interchanging the roles of W and B, we obtain $|W' \cap N(T)| \leq |N(T)| - |T|$. Denote $W - S - N(T) = A$. Using the facts that $|B'| = |W'|$ and $|B| = |W|$, we have

$$\begin{aligned}
|B' \cap N(S)| &\leq |B'| = |W'| = |W' \cap (N(T) \cup A)| \\
&= |W' \cap N(T)| + |A| \\
&\leq |N(T)| - |T| + |W - S - N(T)| \\
&= |N(T)| - |B - N(S)| + |W| - |S| - |N(T)| \\
&= |N(S)| - |S|.
\end{aligned}$$

We have shown (4.1) for any $S \subseteq W$. ■

Bibliography

[1] R.E.L. Aldred, R.P. Anstee and S.C. Locke, Perfect matchings after vertex deletions, Preprint.

[2] R.P. Anstee, A polynomial algorithm for b-matching: An alternative approach, *Information Processing Letters* 24 (1987), 153-157.

[3] R.P. Anstee, Matching theory: fractional to integral, *New Zealand J. Math.* 21 (1991), 17-32.

[4] T. Asano, M. Edahiro, H. Imai, M. Iri. and K. Murota, Practical use of bucketing techniques in computational geometry, *Computational Geometry* (1985), 153-195.

[5] C. Berge, Two theorems in graph theory, *Proc. Nat. Acad. Sci. U.S.A.* 43 (1957), 842-844.

[6] U. Derigs, Programming in networks and graphs, *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag, 300 (1988).

[7] E.W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.* 1 (1959), 269-271.

[8] J. Edmonds, An introduction to matching, Lecture notes, University of Michigan, Ann Arbor (1967).

[9] J. Edmonds and E. Johnson, Matchings, Euler tours, and the Chinese postman, *Math. Programming* 5 (1973), 88-124.

[10] E. Egerváry, On combinatorial properties of matrices, *Math. Lapok* 38 (1931), 16-28.

[11] P. Elias, A. Feinstein and C.E. Shannon, Note on maximum flow through a network, *IRE Trans. on Information Theory* IT-2 (1956) 117-119.

[12] L.R. Jr. Ford and D.R. Fulkerson, Maximal flow through a network, *Canad. J. Math.* 8 (1956) 399-404.

[13] R.H. Fowler and G.S. Rushbrooke, *Trans. Faraday Soc.* 33 (1937), 1272.

[14] D.R. Fulkerson, A.J. Hoffman and M.H. McAndrew, Some properties of graphs with multiple edges, *Canad. J. Math.* 17 (1965), 166-177.

[15] M. Guan, Graphic programming using odd and even points, *Chinese Math.* 1 (1962), 273-277.

[16] P. Hall, On representation of subsets, *J. Lond. Math. Soc.* 10 (1935), 26-30.

[17] R.E. Jamison, N. Lochner, Tiling fringed chessboards with dominoes, 34th Southeastern International Conference on Combinatorics, Graph Theory and Computing.

[18] P.W. Kasteleyn, The statistics of dimers on a lattice, *Physica* 27 (1961), 1209-1225.

[19] P.W. Kasteleyn, Graph theory and crystal physics, *Graph Theory and Theoretical Physics*, Academcis Press, New York (1967), 43-110.

[20] D. König, Graphen und matrizen, *Math. Lapok* 38 (1931) 116-119.

[21] H.W. Kuhn, The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955), 83-97.

[22] S.M. Lee and J.P. Shim, *Micro Management Science*, Allyn and Bacon, second edition, 1990, 218-219.

[23] K. Menger, Zur allgemeinen Kurventheorie, *Fund. Math.* 10 (1927), 95-115.

[24] D.L. Miller and J.F. Pekny, A staged primal-dual algorithm for perfect *b*-matching with edge capacities, *ORSA J. of Computing* 7 (1995), 298-320.

[25] J. Munkres, Algorithms for the assignment and transportation problems, *J. Soc. Indust. Appl. Math.* 5 (1957), 32-38.

[26] M. Müller-Hannemann, *Quadrilateral Mesh Generation in Computer-Aided Design*, Cuvillier Verlag Göttingen, 1997.

[27] J. Petersen, Die theorie der regulären graphen, *Acta Math.* 15 (1891), 193-220.

[28] W.R. Pulleyblank, Faces of matching polyhedra, Ph.D. thesis, Faculty of Mathematics, University of Waterloo (1973).

[29] J.K. Roberts, *Proc. Roy. Soc. (London)* A, 161 (1935), 141.

[30] W.T. Tutte, The factorization of linear graphs, *J. Lond. Math. Soc.* 22 (1947), 107-111.

[31] D.B. West, *Introduction to Graph Theory*, Prentice Hall, 1996.