On Linear Programming Relaxations of Hypergraph Matching

CHAN, Yuk Hei

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2009

Abstract

On Linear Programming Relaxations of

Hypergraph Matching

CHAN, Yuk Hei

Master of Philosophy

Department of Computer Science and Engineering

The Chinese University of Hong Kong

2009

A hypergraph is a generalization of a graph where each hyperedge can contain an arbitrary number of vertices. The hypergraph matching problem is to find a largest collection of disjoint hyperedges. While matching on general graphs is polynomial time solvable, hypergraph matching is NP-hard and there is no good approximation algorithm for the problem in its most general form. We study the restricted case where every hyperedge consists of k vertices, known as k-Set Packing.

We show that the integrality gap (the maximum ratio of the fractional solution to the integral solution over all hypergraphs) is $k - 1 + \frac{1}{k}$, and that the integrality gap analysis is tight in general. The proof is by showing a fractional colouring with a small number of colours. The colouring is done in a greedy manner with the help of a good ordering of hyperedges, which is derived from the LP solution.

As a special case, we prove that by adding a *Fano plane constraint* which deals with a set of intersecting hyperedges, the integrality gap of the LP for unweighted 3-Set Packing can be improved from $\frac{7}{3}$ to 2. The result is by detailed analysis on the structure of a counterexample H, where we show that there is no Fano plane in H and by a result of Füredi [15], we conclude that the integrality gap is 2.

When the vertex set is partitioned into k sets so that each hyperedge contains one vertex from each set, we have the k-Dimensional Matching problem. In this case we show that the standard LP relaxation has an integrality gap of $k - 1$. We remark that integrality gap analysis had been made by Füredi, Kahn and Seymour [16] in a more general form using a different approach, but their analysis does not directly yield an approximation algorithm. We obtain an $(k - 1)$-approximation algorithm by combining the good ordering on the hyperedges with an algorithmic framework called *local-ratio*. This improves the previous result for 3-Dimensional Matching from $2 + \epsilon$ to 2.

摘要

On Linear Programming Relaxations of

Hypergraph Matching

關於超圖的線性規劃鬆弛

陳旭熙

香港中文大學

計算機科學與工程學系

哲學碩士

二零零九

超圖 (hypergraph) 是圖的一種推展，當中每條超邊 (hyperedge) 可以包含任意數量的節點。超圖匹配 (hypergraph matching) 問題要求一組最大不相交的超邊。雖然圖的匹配 (graph matching) 是多項式時間可解，但超圖匹配是 NP-困難 (NP-hard) 的，並且在最整體的情況下沒有好的近似算法 (approximation algorithm)。本文研究當每條超邊包含 k 個節點時的情況。此問題又稱為 k 集包裝 (k-Set Packing)，是超圖匹配問題的其中一種情況。

我們證明 k 集包裝的最優解之差 (integrality gap; 在所有超圖中分數解與整數解的最大比率) 為 $k-1+\frac{1}{k}$，而且這分析一般來說是緊密 (tight) 的。這個證明用了分數着色 (fractional colouring)：我們展示一種用少量顏色把圖分數着色的方法。我們用線性規劃 (LP) 的解得出一個超邊的次序，然後依此次序用貪婪法把圖着色。

在其中一個特殊情形 (非帶權 3 集包裝問題)，我們證明對於每組相交的超邊加上 *Fano* 平面約束 (Fano plane constraint)，可以把線性規劃鬆弛 (LP relaxation) 的最優解之差從 $\frac{7}{3}$ 改善為 2。這一結果是對於一個反例圖 H 作出詳細分析得出的。我們證明 H 並不包含 Fano 平面，然後用 Füredi [15] 的一個定理，得出最優解之差為 2 的結論。

如果所有節點被分成 k 個集合，使得每條超邊都包含每個集的一個節點，這個匹配問題就是 k 維匹配 (k-Dimensional Matching) 問題。在這個情況下，我們證明標準線性規劃鬆弛的最優解之差為 $k-1$。我們注意到 Füredi、Kahn 及 Seymour [16] 亦曾對這個問題用另一種方法進行分析，但是他們的分析並不能直接轉化成近似算法。我們用上述的次序及一個名為 *local-ratio* 的算法框架，得出一個近似比率為 $k-1$ 的算法，把 3 維匹配算法的近似比率從 $2+\epsilon$ 改進為 2。

Acknowledgements

First of all, my deepest gratitude goes to Professor Lap Chi Lau, my supervisor, for his constant guidance and encouragement. He spent a lot of time discussing with me the problem I am working on and gave valuable suggestions. Without his detailed advice this thesis could not have been completed.

Second, I would like to thank my colleagues Darek Yung, Isaac Fung and Jesse Zhang for their advices and helps. Whenever I have something in mind, they are happy to listen and discuss with me. They made my life in Rm. 115 an enjoyable one.

I would also like to thank the professors in the theory group for organizing and attending the theory lunch and reading group, which I enjoyed and learned a lot.

Finally, I would like to express my gratitude to my family for their love and support. They are always there to make sure I have what I need. It is to them that I dedicate this work.

Contents

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

1.1.1 Hypergraph Matching

Before we go into the problem definition, we first introduce *hypergraphs*. A hypergraph is a generalization of an ordinary graph and problems we are discussing can be formulated in terms of matching in certain classes of hypergraphs.

Formally, a hypergraph $H = (V, E)$ consists of a set V of vertices and a set E of hyperedges where each hyperedge contains any number of vertices from V besides zero (i.e. it cannot be empty). If every hyperedge contains 2 vertices, then it is equivalent to an ordinary graph. As opposed to ordinary graphs, hypergraphs may not be easy to visualize since every edge contains an arbitrary number of vertices. Each hyperedge can be treated as a non-empty subset of V so that operations on hyperedges can be denoted by set notation.

In the remainder of the thesis, we use "edge" and "hyperedge" interchangeably, as well as "graph" and "hypergraph" interchangeably when no confusion may arise.

Definition (Hypergraph matching)**.** *Given a hypergraph $G = (V, E)$, a matching on G is a subset $M \subseteq E$ of hyperedges such that for any distinct $e_i, e_j \in M$, $e_i \cap e_j = \emptyset$.*

In unweighted case, we find a matching that maximizes the cardinality (number of edges) while in weighted case, each hyperedge is associated with a weight and we maximize the weighted sum of the matching. This problem is equivalent to set packing, as well as maximum independent set. If we don't impose any restrictions on the hypergraph, there is a $\sqrt{n}$ approximation algorithm [21, 20], where $n = |V|$. Assuming NP $\neq$ ZPP, the result by Håstad [19] implies that this is the best possible. To obtain better approximation ratio we restrict ourselves to two restricted classes of hypergraphs.

Here are some notations that we will be using. The degree $\deg(v)$ of a vertex v in a graph is the number of edges containing v. A graph is called k-regular if the degree of every vertex is k. It is called a k-uniform hypergraph if every edge e in the graph contains exactly k vertices. Finally, a graph is called k-partite if the set of vertices can be partitioned into k disjoint set, such that every edge contains one vertex from each set. From the definition, a k-partite hypergraph is automatically a k-uniform hypergraph. We use the notation $\delta(v)$ to denote the set of edges incident to a vertex v and $E(S)$ to denote the set of edges that have both end points in a vertex set $S \subseteq V$.

1.1.2 k-Set Packing

The problem of k-Set Packing (k-SP in short) is defined as follows: Given a ground set V and a collection $\mathcal{S}$ of sets, each of them contains k element from V, find a maximum subcollection of sets so that they are pairwise disjoint. The case $k = 2$ is equivalent to matching in general graphs and is polynomial time solvable. Since it is a generalization of k-Dimensional Matching (see below), the problem is NP-complete for $k \geq 3$. In terms of hypergraph, k-Set Packing is the problem of finding a maximum matching on a k-uniform hypergraph.

In the weighted variant of the problem, one is asked to find a collection of disjoint sets that has the maximum total weight instead of maximum cardinality.

1.1.3 k-Dimensional Matching

The problem of k-Dimensional Matching (k-D Matching or k-DM in short) is defined as follows: given k disjoint vertex sets $V_1, V_2, \ldots, V_k$ and a set $\mathcal{S}$ of tuples (hyperedges), each of which contains one vertex from each V_i, find a maximum set of tuples that are pairwise disjoint. It is a special case of k-Set Packing. For the case $k = 2$, this is the bipartite matching problem, which can be solved in polynomial time by a number of algorithms, including the Hopcroft-Karp algorithm and the Hungarian algorithm. For $k \geq 3$, the problem becomes NP-complete. 3-Dimensional Matching, in particular, is one of the famous *Karp's 21 NP-complete problems*. In terms of hypergraph, k-D Matching asks to find a maximum matching in an k-partite hypergraph.

1.1.4 Related Problems

The two problems mentioned above are versatile and many problems can be mapped to either k-DM or k-SP.

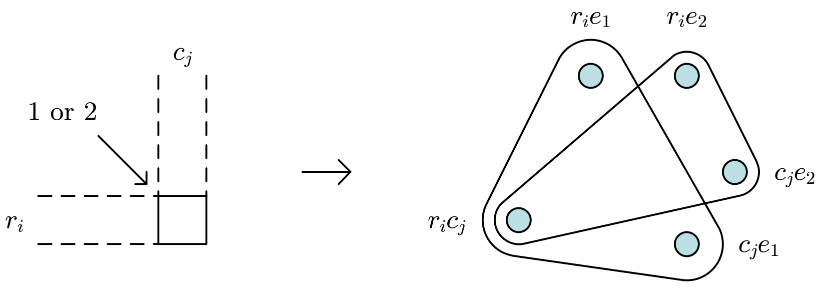


Figure 1.1: If an empty cell in row i, column j can be filled with colour 1 or 2 (left), there are 2 hyperedges corresponding to this cell (right).

Partial Latin Square Extension Problem

The partial Latin square extension problem (PLSE) is defined as follows. Given an $n \times n$ array of squares with some squares already filled with colour $1, \ldots, n$, find a completion that fills as many empty squares as possible, so that no rows or columns contain a colour more than once.

We state an approximation ratio-preserving reduction to the 3-DM problem. We have n^2 vertices for each partition. Vertices in the different part are labelled r_ic_j, r_ie_k and c_je_k respectively. Call a vertex r_ic_j *active* if the square in row i, column j is empty. If row i does not contain colour k then r_ie_k is active, and we do the same for vertices c_je_k. If all three vertices r_ic_j, r_ie_k and c_je_k are active, then we have a hyperedge containing them (Figure 1.1). Then we solve the instance of 3-DM. If a hyperedge (r_ic_j, r_ie_k, c_je_k) is selected, the cell in row i, column j is filled with colour k. By the way the hyperedges is defined, we can see that (i) each cell is filled with at most one colour and (ii) each colour is used at most once in each row and column, so the matching corresponds to a legal completion. Since each hyperedge one-to-one corresponds to a possible colouring of a cell, the approximation ratio is preserved.

Sudoku Extension Problem

Following the same idea as in the previous section, we can map the Sudoku extension problem to 4-DM. Sudoku, in its most popular form, is a 9×9 Latin square with the additional constraint that each colour should appear once in each of the 3×3 "box". To fill in as much squares as possible using an approximation algorithm, first construct the 3-partite hypergraph with hyperedges (r_ic_j, r_ie_k, c_je_k) ignoring the box constraint. Then, add the fourth dimension with vertices e_kb_l indicating that colour k is not found in box l. For a hyperedge (r_ic_j, r_ie_k, c_je_k), extend it to the fourth dimension with vertex e_kb_l if the cell (i, j) is in box l. The set of hyperedges of size 4 form an instance of 4-DM so that we colour cell (i, j) by k satisfying the box constraint when the hyperedge

$(r_i c_j, r_i e_k, c_j e_k, e_k b_l)$ is selected.

Maximum Independent Set on Degree Bounded Graphs

The maximum independent set (MIS) problem on degree bounded graphs can be mapped to k-Set Packing when the degree bound is k. Each vertex in the graph is mapped to a set in the packing problem and each edge is mapped to a distinct element in the ground set. A set (vertex) contains all elements (edges) incident to it. Since degree of a vertex is bounded by k, there are at most k elements in each set. If the degree of a vertex is less than k we can add dummy elements to make it size k. Therefore MIS on degree bounded graphs is a special case of k-SP.

Maximum Independent Set on $(k + 1)$-claw-free Graphs

The k-Set Packing problem is sometimes studied under a more general problem of maximum independent set (MIS) on $(k + 1)$-claw-free graphs. A claw is a complete bipartite graph $K_{1,3}$. A claw-free graph does not contain a claw as an induced subgraph. Finding a maximum independent set on claw-free graph is a generalization of the general matching (2-Set Packing problem) and is solvable in polynomial time [27, 28]. Consider the intersection graph of the edges: if two edges share the same endpoint, the corresponding vertices is connected by an edge in the intersection graph. This graph is claw-free because at most two edges in the neighbourhood of each edge can appear in each matching.

A k-claw, which is a generalization of claw, consists of a *center* node connected to k vertices (*talons*) that form an independent set (a claw is equivalent to a 3-claw under this notation); it is also known as the complete bipartite graph $K_{1,k}$ or a star. To map an instance of k-SP to MIS on $(k + 1)$-claw-free graph, just take the intersection graph of the sets. An independent set on the intersection graph corresponds to a valid set packing since no two sets intersect. The graph is $(k + 1)$-claw-free because each set s consists of k elements so there cannot be an independent set of size larger than k in the neighbourhood of s in the intersection graph.

To see that MIS on $(k + 1)$-claw-free graphs is more general than k-SP, consider the wheel graph W_{2k+2} formed by adding a universal vertex to the cycle C_{2k+1} (Figure 1.2). It is not hard to see that the wheel is $(k + 1)$-claw-free since any independent set is of size at most k. However, there is no configuration of k-sets that would result in this intersection graph: the largest complete subgraph in the intersection graph is a triangle, which covers two consecutive elements on the cycle and the center. Disjoint triangles must use different elements as otherwise extra edges would appear in the intersection

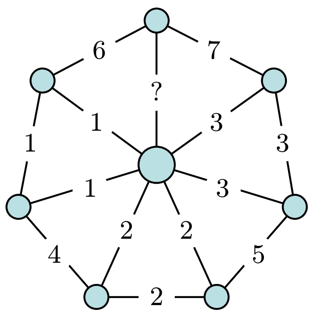


Figure 1.2: The wheel graph W_8 does not correspond to any 3-SP instance (number on the edge indicates the element shared).

graph. Hence we come to the conclusion that such a wheel does not correspond to any k-SP instance.

1.2 Main Result

In this work, we study the standard linear program relaxation for the k-Dimensional Matching (we use the shorthand $x(S)$ to denote $\sum_{x \in S} x_e$ for a subset of hyperedges $S \subseteq E$):

$$\text{(LP)} \qquad \begin{aligned} \max \quad & \sum_{e \in E} w_e x_e & \\ \text{s.t.} \quad & x(\delta(v)) \leq 1 & \forall v \in V \\ & x_e \geq 0 & \forall e \in E \end{aligned}$$

We show that the integrality gap is $(k-1)$ and give an approximation algorithm for the weighted problem. This yields an improved approximation algorithm for $k = 3$.

Theorem 1.1. *The integrality gap of* (LP) *is at most $k-1$.*

Theorem 1.2. *There is a polynomial time $(k-1)$-approximation algorithm for the weighted k-Dimensional Matching problem.*

The algorithm is based on iterative rounding (Section 2.3) and local ratio (Section 2.5). We look at the basic solution of the linear program to construct an ordering of edges, and then use local ratio to obtain an approximate solution. Details of the technique used will be presented in Chapter 3.

After that, we move on to k-Set Packing. The integrality gap in this case is not $k-1$, as can be seen from the *Fano plane* (Figure 1.3): if we set $x_e = \frac{1}{3}$ for all edge, it is a valid solution to the linear program having a weight of $\frac{7}{3}$ (if weight of every edge is 1). The integral solution is 1 because the graph is intersecting (every edge intersects all

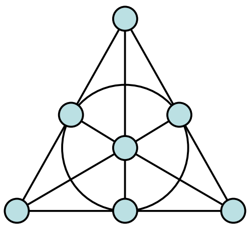


Figure 1.3: Fano plane as an integrality gap example

other edges). Therefore the integrality gap is at least $\frac{7}{3}$. In general, the *projective plane* (Section 2.4.1) serves as an example that the integrality gap for the k-Set Packing linear program is no less than $k-1+\frac{1}{k}$.

Theorem 1.3. *The integrality gap of* (LP) *for k-uniform hypergraph is at most $k-1+\frac{1}{k}$.*

We then ask the question of how to improve the linear program. From the example above we see that projective planes are bad for the linear program. For $k=3$ we write the Fano plane constraint which states that sum of x_e in a Fano plane must not exceed 1 (because it is intersecting):

$$x(P) \leq 1 \qquad \forall \text{ Fano plane } P.$$

Call the resulting linear program by Fano linear program, denoted by *Fano-LP*. We show that the integrality gap of the Fano linear program for unweighted case is improved to 2, which is the same as that of 3-D Matching.

Theorem 1.4. *The* Fano-LP *for unweighted* 3*-uniform hypergraphs has integrality gap exactly* 2*.*

Consider a minimum counterexample to the theorem. We show, in several steps, that this minimum counterexample does not contain the Fano plane a sub-hypergraph. By a result of Füredi [15], which states that in the absence of a Fano plane, the integrality gap of (LP) is 2, we obtain the result as claimed.

1.3 Overview of the Thesis

In Chapter 2, we introduce background materials related to this thesis, including matching in ordinary graphs, linear programming, previous work on the problem using local search and the local ratio method that we use in the approximation algorithm.

In Chapter 3, we prove that the integrality gap of the standard linear programming relaxation for k-Dimensional Matching is $k-1$. Then we show a $(k-1)$-approximation algorithm for the weighted k-D Matching. For the case $k = 3$, previous results [3, 7, 8] are $(2+\epsilon)$-approximate where ϵ depends on the running time one is willing to afford (although their algorithms work for a more general setting of k-Set Packing). Our algorithm is 2-approximate and the running time does not depend on the approximation ratio. For larger value of k, the tight integrality gap of $k - 1$ means that algorithms that based solely on this linear program cannot compete with local search algorithms.

In Chapter 4, we show that the argument in the previous chapter can be used to show that the integrality gap of the standard linear programming relaxation of is k-Set Packing is $k - 1 + \frac{1}{k}$. By adding a type of constraint which we call Fano plane constraint, we then show in the case $k = 3$, the integrality gap for the unweighted linear program can be improved to 2, which is the same as the case for weighted 3-D Matching.

Chapter 2

Background

2.1 Matching

2-D Matching and 2-Set Packing are equivalent to matching in bipartite graphs and general graphs respectively, and they are important combinatorial problems that can be solved in polynomial time. Two of the methods to solve these problems include augmenting paths and linear programming. Both techniques can be extended to obtain approximation algorithms for k-DM and k-SP. In this section, we review what we know about matching.

Maximum matching on general graphs is defined as follows: given a graph $G = (V, E)$, find a subset of edges no two of them share the same vertex. If the graph is bipartite, we have the maximum bipartite matching problem. The minimum vertex cover problem asks for a minimum set of vertices that cover all the edges, that is, each edge should have at least one endpoint in the cover.

König's theorem [25] states that in a bipartite graph, the cardinality of a minimum vertex cover is the same as that of a maximum matching. Such exact min-max relation does not hold for general graphs, as can be seen from a complete graph on 3 vertices: minimum vertex cover is 2 while maximum matching is 1.

2.1.1 Augmenting Path

In this section, we define what is an augmenting path and describe the augmenting path algorithm for bipartite matching. Also, we show that if there is no short augmenting path, then the solution we get is close to optimal.

Given any matching $M \subset E$ (not necessarily maximum), a vertex is called *free* if it is not incident to any edges in M. An augmenting path is a path that starts an edge outside the matching which is incident to a free vertex, alternates between edges inside

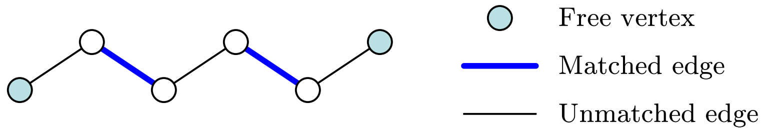


Figure 2.1: An augmenting path

```
Algorithm AugmentingPath
Input: A bipartite graph $G = (U \cup V, E)$, a matching $M$
Output: A matching $M'$ with $|M'| > |M|$, if one exists.
For each edge $e = (u, v)$ in the graph where $u \in U$, $v \in V$
    If $e$ is in the matching $M$ then
        orient the edge $e$ as $v \to u$;
    else orient the edge $e$ as $u \to v$;
If there is a directed path $P$ from a free vertex $u \in U$ to $v \in V$ then
    return $M' = (M - (P \cap M)) \cup (P - M)$;
else return "No larger matching exists";
```

Figure 2.2: Augmenting path algorithm

and outside the matching, and ends with an edge outside the matching incident to a free vertex (Figure 2.1). If there is an augmenting path P, then by removing $P \cap M$ from the matching and adding $P - M$ to the matching, the size of the matching is increased by one. It can be shown that a matching is maximum if and only if there is no augmenting path (Chapter 16 of [30]).

Figure 2.2 shows the augmenting path algorithm for maximum bipartite matching. The process is repeated until no larger matching can be found, in that case the matching is maximum.

While the matching is maximum if there is no augmenting path, we can say something about the solution when it does not contain short augmenting paths of length at most $2t + 1$. Let Opt be a fixed optimal solution and Apx be a solution without augmenting paths of length most $2t + 1$. Then Opt $\leq (1 + \frac{1}{t})$Apx [3]. First, add zero weight edges to the graph to make it a complete bipartite graph. Then, Opt $\cup$ Apx consists of edges of Opt $\cap$ Apx and cycles with edges alternating between Opt and Apx. In each such cycle, the weight of edges in Opt is at least that of Apx edges. Consider a cycle where the weight of Opt edges equals that of Apx edges. Let edges in the cycle be $A_0, O_0, A_1, O_1, \ldots, A_{m-1}, O_{m-1}$, where $A_i \in$ Apx and $O_i \in$ Opt. Since there are no short augmenting paths, $m \geq t + 1$, and $w(A_i) + \cdots + w(A_{i+t}) \geq w(O_i) + \cdots + w(O_{i+t-1})$ (where indices are taken modulo m). By summing the inequality for $i = 0, \ldots, m - 1$, each of A_i is taken $t + 1$ times and each of O_i is taken t times, so $(t + 1)$Apx $\geq t \cdot$ Opt.

2.1.2 Linear Programming

Linear Programming (LP) solves an optimization problem with linear objective function, subject to linear equalities and inequalities.

In matrix form, a linear program (for maximization) can be written as

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leq \mathbf{b}, \qquad \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

where A is a matrix of coefficients, $\mathbf{b}$ and $\mathbf{c}$ are vectors of coefficients. The *dual* of the above linear program (*primal* program) is of the form

$$\begin{aligned} \min \quad & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} \quad & A^T\mathbf{y} \geq \mathbf{c}, \qquad \mathbf{y} \geq \mathbf{0}. \end{aligned}$$

Any feasible primal solution has a value greater than or equal to that of any feasible dual solution. The strong duality theorem states that if the primal program has an optimal solution, so does the dual and they have the same objective value ($\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$). Linear Programming is useful in modelling a number of combinatorial optimization problems.

The LP for the maximum bipartite matching can be written as follows:

$$\begin{aligned} \max \quad & \sum_{e \in E} x_e \\ \text{s.t.} \quad & \sum_{e \ni v} x_e \leq 1 \qquad \forall\, v \in V \\ & 0 \leq x_e \leq 1 \qquad \forall\, e \in E \end{aligned}$$

In this problem, we have a variable x_e corresponding to each edge e, which takes on a value of 1 if it is in the maximum matching, 0 otherwise. The discrete constraint is relaxed to $0 \leq x_e \leq 1$ in the linear program. The dual program can be written as:

$$\begin{aligned} \min \quad & \sum_{v \in V} y_v \\ \text{s.t.} \quad & y_u + y_v \geq 1 \qquad \forall\, (u, v) \in E \\ & 0 \leq y_v \leq 1 \qquad \forall\, v \in V \end{aligned}$$

The dual program is exactly the LP for vertex cover. There is one variable y_v for each vertex v. König's theorem implies that for bipartite graphs, both the primal (maximum matching) and the dual (vertex cover) have an integral optimal solution.

2.1.3 Matching in General Graphs

In general graphs, the argument of augmenting paths still hold; however the augmenting path algorithm does not work because there may be odd-length cycles and, depending on how the cycle is traversed, we may or may not find the augmenting path we want. This problem is solved in Edmonds' famous paper "Paths, Trees, and Flowers" [13] where odd-length cycle, when found, is shrunk into a blossom and treated as if it is a single vertex. The search then continues until an alternating path is found. He proved that we can unshrink the blossoms to give an alternating path in the original graph and therefore, a maximum matching in general graphs can be found in polynomial time.

While the LP for maximum matching above does not yield an integral optimal solution, Edmonds also showed that by adding a type of constraint called odd-set inequalities, we can obtain an integral optimal solution [14]. The odd-set constraint states that for each odd subset S of vertices, we allow at most $\lfloor |S|/2 \rfloor$ edges to be chosen ($E(S)$ denotes the set of edges with both ends in S):

$$\begin{array}{lll}
\max & \sum_{e \in E} x_e & \\
\text{s.t.} & \sum_{e \ni v} x_e \le 1 & \forall\, v \in V \\
& \sum_{e \in E(S)} x_e \le \left\lfloor \frac{|S|}{2} \right\rfloor & \forall\, S \subseteq V, |S| \text{ is odd} \\
& 0 \le x_e \le 1 & \forall\, e \in E
\end{array}$$

There are exponentially many such constraints, but it can be solved in polynomial time using ellipsoid method with a separation oracle.

2.1.4 Approximate Min-max Relation for Hypergraphs

The min-max relation does not hold in the generalized setting of k-partite hypergraphs, as can be seen from this example: consider the 3-partite hypergraph with edges (a_0, b_2, a_1), (a_1, b_0, a_2), (a_2, b_1, a_0) (where a_i and b_i belong to the same partition for $i = 0, 1, 2$). The maximum matching is 1 because every pair of edges share one vertex, while the minimum vertex cover (transversal) is 2 by choosing any two of a_i's. Although the exact min-max does not hold, we have an approximate min-max relation for k-dimensional matching and vertex cover.

Denote the size of a maximum matching by ν and the size of a minimum vertex cover by τ. It is not difficult to see that in a k-uniform hypergraph, $\tau \le k\nu$ because all vertices in a maximum matching must form a cover. For k-partite hypergraphs, it is conjectured that a tighter bound applies:

Conjecture (Ryser)**.** *In a k-partite hypergraph, $\tau \leq (k-1)\nu$.*

In bipartite graphs ($k = 2$), Ryser's conjecture reduces to König's theorem. The case $k = 3$ is proved by Aharoni [1] using an extension of Hall's theorem for hypergraphs [2], whose proof is topological. The case $k \geq 4$ is still open.

2.2 Local Search

Local Search is a heuristic to find a good approximate solution by means of *local moves*. It can be viewed as an extension of the augmenting path argument (Section 2.1.1) to hypergraphs. Almost all known algorithms for approximating k-Set Packing are based on local search.

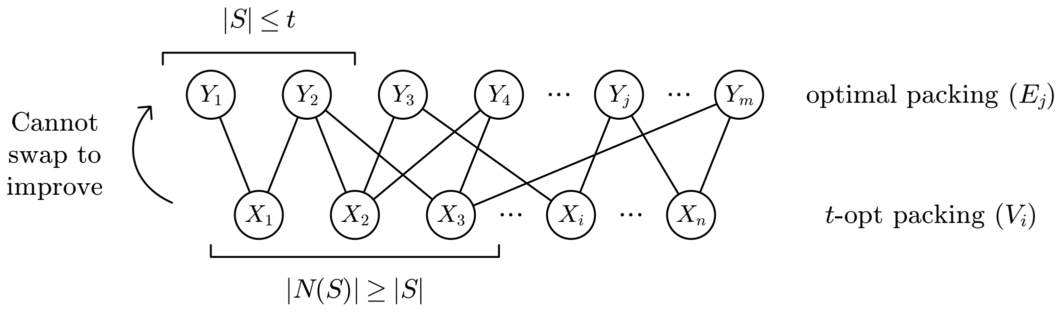
Take the unweighted k-Set Packing as an example. A local search algorithm for the problem will look for $s \leq t$ pairwise disjoint sets outside the current solution set X that intersect at most $s-1$ sets in X. If such a collection S of s sets is found, we can improve our solution by removing all the sets in X that intersect any set in S (at most $s-1$ of them) and add S to X. This procedure is repeated until no improvement can be made. The solution is called t-optimal (t-opt). In particular, a 2-opt solution is a solution that we cannot improve by swapping one k-set in the solution for two k-sets not in the solution. The move is *local* because it only focus on small collections S where $|S| \leq t$ and t is independent of n.

In this section, we first look at the local search algorithm for the unweighted k-SP and then a number of algorithms for the weighted case. We highlight the method and proof idea they use.

2.2.1 Unweighted k-Set Packing

Hurkens and Schrijver [23] proved that if we perform local search on the unweighted k-Set Packing, we get a $(\frac{k}{2} + \epsilon)$-approximation algorithm; in particular, a 2-opt solution is $\frac{k+1}{2}$-approximate. They obtained this result by studying the following problem:

> Given a ground set V of size n and a number of sets $E_1, E_2, \ldots, E_m$, such that (i) each element of V appears in at most k of the sets and (ii) for each collection of at most t sets from $\{E_i\}$, there is a system of distinct representatives (SDR)[1]. How large can the ratio $\frac{m}{n}$ be?



$|S| \leq t$
Cannot swap to improve
Y_1 Y_2 Y_3 Y_4 $\cdots$ Y_j $\cdots$ Y_m
optimal packing (E_j)
X_1 X_2 X_3 $\cdots$ X_i $\cdots$ X_n
t-opt packing (V_i)
$|N(S)| \geq |S|$

Figure 2.3: Setting as discussed in [23]

Theorem 2.1 ([23]). *In an instance satisfying conditions* (i) *and* (ii)*, the ratio* $\frac{m}{n}$ *satisfies*

$$\frac{m}{n} \leq \frac{k(k-1)^r - k}{2(k-1)^r - k} \qquad \text{if } t = 2r-1;$$
$$\frac{m}{n} \leq \frac{k(k-1)^r - 2}{2(k-1)^r - 2} \qquad \text{if } t = 2r.$$

Note that by König-Hall theorem, condition (ii) is equivalent to saying that for any collection of $s \leq t$ sets from $\{E_i\}$, the union contains at least s elements. Consider an instance of the k-Set Packing problem where we obtain a t-opt solution $X_1, X_2, \ldots, X_n$. Let the optimal packing be $Y_1, Y_2, \ldots, Y_m$. We claim that the ratio $\frac{m}{n}$ can only be as large as that in the problem above. This can be seen by treating $\{X_i\}$ in our t-opt solution as the elements of the ground set, and $\{Y_j\}$ in the optimal solution to be the set E_j in the above problem. Y_j contains X_i iff $X_i \cap Y_j \neq \emptyset$. Every element X_i appears in at most k of the sets Y_j since $|X_i| = k$ and $\{Y_j\}$ is a packing. From the fact that $\{X_i\}$ it t-optimal, for every collection of $s \leq t$ sets from $\{Y_j\}$, the union contains at least s sets X_i (i.e. there is an SDR) (Figure 2.3). Therefore the conditions are satisfied and the performance ratio is given by Theorem 2.1. The authors also gave an example showing the bounds given in the theorem is tight.

The proof of Theorem 2.1 is by induction on t and we will not go into the details. We only give the proof for the case $t = 2$, which will establish the fact that a 2-opt solution is $\frac{k+1}{2}$-approximate. Without loss of generality, let the singletons be $E_{h+1}, E_{h+2}, \ldots, E_m$ for some $h \leq m$. Since $t = 2$, no two singletons share the same element so $m - h \leq n$. Also,

$$m + h = 2h + (m-h) \leq \sum_{i=1}^{h} |E_i| + \sum_{i=h+1}^{m} |E_i| = \sum_{i=1}^{m} |E_i| \leq kn.$$

Therefore $2m = (m-h) + (m+h) \leq (k+1)n$, i.e. $\frac{m}{n} \leq \frac{k+1}{2}$.

[1]An SDR for a collection of sets $\{E_i\}$ is a set $\{v_i\}$ such that $v_i \in E_i$ for all i and $v_i \neq v_j$ for $i \neq j$.

2.2.2 Weighted k-Set Packing — $(k-1+\epsilon)$-approximation

The situation in weighted case is more complicated. Arkin and Hassin [3] studied the weighted problem. Given a matching M and a set R of k-sets, let $N_M(R)$ be the set of k-sets in M that has non-empty intersection with any k-set in R. The local search algorithm they analyzed is as follows: for each subcollection R of $|R| \leq t$ pairwise disjoint sets outside the current matching M, see if the weight of sets in R is greater than the weight of sets in $N_M(R)$. If so, we replace $N_M(R)$ by R to get a matching of larger weight. Note that $|R|$ may be smaller that $|N_M(R)|$.

To model the t-opt solution and analyze the performance ratio of the algorithm, they use a similar approach as in the previous section. Specifically, they look at a weighted bipartite graph $G = (U, V; E)$, where U denotes the collection of sets in the optimal packing and V denotes the t-opt solution. A t-opt solution in this case is that, for every subset $R \subseteq U$ with $|R| \leq t$, the sum of weights in the neighbourhood $w(\cup_{u \in R} N(u))$ is at least $w(R)$. They obtain the worst possible ratio $\frac{w(U)}{w(V)}$ by writing the following LP:

$$\begin{array}{lll}
\max & \displaystyle\sum_{u \in U} w_u & \\
\text{s.t.} & \displaystyle\sum_{v \in V} w_v = 1 & \\
& \displaystyle\sum_{u \in R} w_u \leq \sum_{v \in \cup_{u \in R} N(u)} w_v & \forall\, R \subseteq U, |R| \leq t \\
& w_i \geq 0 & i \in U \cup V.
\end{array}$$

The dual program is:

$$\begin{array}{lll}
\min & z & \\
\text{s.t.} & \displaystyle\sum_{R \ni u} y_R \geq 1 & \forall\, u \in U \\
& z \geq \displaystyle\sum_{R: R \cap N(v) \neq \emptyset} y_R & \forall\, v \in V \\
& y_R \geq 0 & \forall R \subseteq U.
\end{array}$$

They bound the ratio by showing a feasible dual solution of value at most $k-1+\frac{1}{t}$. Instead of working on an arbitrary instance, they transform an instance to a new instance with high girth ($> 2t$) while keeping the ratio $\frac{w(U)}{w(V)}$. Also, by adding dummy nodes, they assume the graph to be k-regular. The instance now looks "symmetric" and this allows us to assign a constant (depending on t and k only) to each y_R to form a feasible dual solution.

The authors also showed that the analysis is tight. We remark that an independent analysis made by Bafna et al. [4] achieves the same bound.

2.2.3 Weighted k-Set Packing — $(2(k+1)/3+\epsilon)$-approximation

The result in the previous section applies to *any* local optimum solution. Chandra and Halldórsson [11] showed that if we start with a greedy initial solution and then repeatedly take the *best* improvement step, we would end up at a local optimum that is provably better. They obtained an approximation ratio of $2(k+1)/3+\epsilon$.

They considered the problem of weighted independent set on $(k+1)$-claw-free graphs, which is a generalization of the weighted k-Set Packing problem as mentioned in Section 1.1.4. The improvement step they take is the following: let I be an independent set, x be a member of I, and $Q \subseteq N(x)$ be an independent set. The new independent set I' is formed by adding Q to I and removing vertices from I which are adjacent to any vertices in Q: $I' = (I \cup Q) - N_I(Q)$.

The goodness of an improvement is measured by the *payoff factor*, defined as $\dfrac{w(Q)}{w(N_I(Q))}$, the weight of vertices swapped in divided by weight of vertices swapped out. Figure 2.4 shows the local search algorithm used.

```
Algorithm BestImp
$I \leftarrow Gr$, the greedy solution;
**while** $I$ is not locally optimal **do**
    Let $I'$ be the improvement of maximum payoff factor;
    $I \leftarrow I'$;
**return** $I$;
```

Figure 2.4: Algorithm BestImp as described in [11]

The BestImp algorithm as shown in Figure 2.4 is not polynomially bounded since the improvement can be arbitrarily small and the improvement sequence can be exponentially long. To make the algorithm run in polynomial time, one can truncate the weight of vertices to integer multiples of $\frac{1}{n^2}w(Gr)$, so that the number of improvements is upper bounded by n^2k (since the initial greedy solution is k-approximate). By truncating, we lose an additive $w(Gr)/n \leq \text{Opt}/n$ and this is where the ϵ comes in.

To prove the performance ratio of the algorithm, they use a potential function (which is related to the weight of the solution) to keep track of the progress. Initially, the potential function has a large value compared to the optimal solution. During the algorithm the potential decreases, and at the same time our solution is improved.

It is shown that the analysis of BestImp is tight.

2.2.4 Weighted k-Set Packing — $((k+1)/2+\epsilon)$-approximation

Local search algorithms discussed above use the sum of weights as the objective, which is quite natural. Berman [7] show that by using the sum of *square* of weights instead, we can achieve a better approximation ratio. He showed a tight approximation ratio of $((k+1)/2+\epsilon)$ and is currently the algorithm with the best approximation ratio. Same as the algorithm in the previous section, this algorithm is concerned with finding a maximum weight independent set in $(k+1)$-claw-free graph.

Denote by T_C the k vertices that forms an independent set in a k-claw C. For convenience, a 1-claw consist of a single vertex (i.e. $C = T_C$). The algorithm SquareImp finds a maximum weight independent set in a $(k+1)$-claw-free graph as follows:

```
Algorithm SquareImp
$I \leftarrow \emptyset$;
**while** there exists a claw $C$ such that $T_C$ improves $w^2(I)$
    $I \leftarrow (I - N_I(T_C)) \cup T_C$;
**return** $I$;
```

Figure 2.5: Algorithm SquareImp as described in [7]

Note that in an iteration of SquareImp, the weight of the solution I may decrease, although $w^2(I)$ improves (as guaranteed by Lemma 2.3). A simple example is given in Figure 2.6.

Similar to BestImp, SquareImp is not polynomially bounded but by the same rounding argument, the number of improvements also can be bounded.

The performance ratio of the algorithm cannot be improved by using some other w^c, as seen in this example: the graph G consist of two vertex sets A and B, with $|A| = k$ and element of B is labelled by subsets of A with 1 or 2 elements; hence $|B| = k+\binom{k}{2} = \frac{k+1}{2}|A|$. For $u \in A$ and $v \in B$, the edge (u, v) exists if and only if $u \in v$. Weight of all vertices are equal. SquareImp may first choose all vertices in A and then stop because there is no improvement that can be made.

Instead of analysing SquareImp directly, the author analysed the following algorithm called WishfulThinking.

While SquareImp search for any claw that improves $w^2(I)$, WishfulThinking focus on a type of claw called *nice claw*. Therefore, the performance ratio of SquareImp cannot be worse than WishfulThinking. On the other hand, since WishfulThinking only focus on nice claws, it cannot make more iterations than SquareImp does. With the following 2 lemmas we can show that SquareImp is an approximation algorithm with performance ratio of $(k+1)/2+\epsilon$.

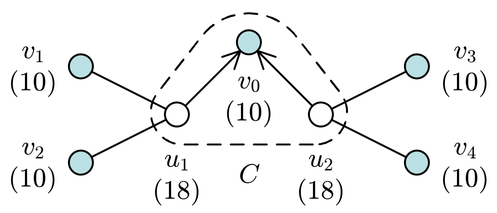


Figure 2.6: By replacing v_i with u_j, the weight decreases.

```
Algorithm WishfulThinking
$I \leftarrow \emptyset$;
while there exists a nice claw $C$
    $I \leftarrow (I - N_I(T_C)) \cup T_C$;
return $I$;
```

Figure 2.7: Algorithm WishfulThinking as described in [7]

Lemma 2.2. *Assume that* WishfulThinking *has terminated and I^* is an independent set. Then $w(I^*)/w(I) \leq (k+1)/2$.*

Lemma 2.3. *If C is a nice claw, then T_C improves $w^2(I)$.*

Running Time

The running time of the algorithm is $\Omega(|V|^k)$. Berman and Krysta [8] showed an approximation algorithm whose running time is $O(l^\alpha(kn)^{2+\alpha})$ for some $1 < \alpha < 2$ and integer $l > 1$. The performance guarantee is at most $2k/3 + \epsilon$. The key for improving the running time is that they focussed on a 2-opt solution with a modified weight function, $\lfloor w^\alpha \rfloor$. The proof involves some calculus and is assisted by a computer program.

2.3 Iterative Rounding

Iterative Rounding is a method for obtaining approximation algorithm devised by Kamal Jain [24]. It involves solving an LP relaxation of the problem to obtain a basic solution, rounding up variables of large value, and solving the residual problem iteratively. For notational simplicity, we use the shorthand $x(S)$ to denote $\sum_{x \in S} x_e$ for a subset of edges $S \subseteq E$.

2.3.1 Basic Solution

The essence of iterative rounding is the use of basic solution. First let us define what is a *vertex*. Let A be an $m \times n$ matrix and $P = \{\, \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \,\} \subseteq \mathbb{R}^n$ be the

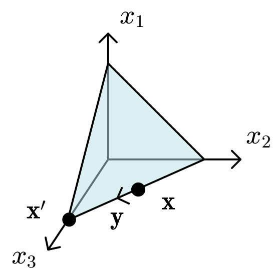


Figure 2.8: $\mathbf{x}'$ is a vertex solution while $\mathbf{x}$ is not since both $\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y} \in P$.

solution space.

Definition. *$\mathbf{x} \in P$ is a* vertex *of P if $\nexists \mathbf{y} \neq \mathbf{0}$ such that $\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y} \in P$.*

As an example, consider $P = \{\, \mathbf{x} \mid x_1 + x_2 + x_3 = 1, \mathbf{x} \geq \mathbf{0} \,\}$. $\mathbf{x}' = (0,0,1)$ is a vertex solution while $\mathbf{x} = (0, 0.5, 0.5)$ is not, because we have $\mathbf{y} = (0, 0.1, -0.1) \neq \mathbf{0}$ such that both $\mathbf{x}+\mathbf{y}$ and $\mathbf{x}-\mathbf{y}$ are in P (Figure 2.8). Geometrically, a vertex is a "corner" of the polytope. The following theorem states that optimal solution is attainable at one of the vertices.

Theorem 2.4. *Assume $\min\{\, \mathbf{c}^T\mathbf{x} \mid \mathbf{x} \in P \,\}$ is finite. Then for all $\mathbf{x} \in P$, there exists a vertex $\mathbf{x}'$ such that $\mathbf{c}^T\mathbf{x}' \leq \mathbf{c}^T\mathbf{x}$.*

In the example above, if we are trying to minimize x_1, then we can "move" from a non vertex optimal solution (say $\mathbf{x}$) to a vertex solution $\mathbf{x}'$ (following the direction of $\mathbf{y}$) while keeping the objective value.

Besides geometric interpretation, we can also check if a solution is a vertex by the use of the following theorem, which provides an algebraic interpretation.

Theorem 2.5. *Let $P = \{\, \mathbf{x} \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0} \,\}$. For $\mathbf{x} \in P$, let $A_{\mathbf{x}}$ denote the submatrix of A by taking columns j such that $x_j > 0$. Then $\mathbf{x}$ is a vertex iff $A_{\mathbf{x}}$ has linearly independent columns (i.e. $A_{\mathbf{x}}$ has full column rank).*

Proof. We prove the contrapositive in both directions. First we prove that if $\mathbf{x}$ is not a vertex, then $A_{\mathbf{x}}$ contains linearly dependent columns. Since $\mathbf{x}$ is not a vertex, there exists $\mathbf{y} \neq \mathbf{0}$ such that $\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y} \in P$. Therefore $A(\mathbf{x}+\mathbf{y}) = \mathbf{b}$, $A(\mathbf{x}-\mathbf{y}) = \mathbf{b}$ so $A\mathbf{y} = \mathbf{0}$. Let $A_{\mathbf{y}}$ be the submatrix of A corresponding to non-zero components of $\mathbf{y}$. Since $A\mathbf{y} = \mathbf{0}$ but $\mathbf{y} \neq \mathbf{0}$, $A_{\mathbf{y}}$ has dependent columns. Moreover, $\mathbf{x}+\mathbf{y} \geq \mathbf{0}$ and $\mathbf{x}-\mathbf{y} \geq \mathbf{0}$ implies that $y_j = 0$ whenever $x_j = 0$. Therefore $A_{\mathbf{y}}$ is a submatrix of $A_{\mathbf{x}}$ and $A_{\mathbf{x}}$ has dependent columns.

In the other direction, suppose $A_{\mathbf{x}}$ has linearly dependent columns. Then there exists $\mathbf{y} \neq \mathbf{0}$ such that $A_{\mathbf{x}}\mathbf{y} = \mathbf{0}$. Extend $\mathbf{y}$ to $\mathbb{R}^n$ by adding zero components. Then there exists $\mathbf{y} \in \mathbb{R}^n$ such that $A\mathbf{y} = \mathbf{0}$, $\mathbf{y} \neq \mathbf{0}$ and $y_j = 0$ whenever $x_j = 0$. We claim that $\mathbf{x} + \mathbf{y}', \mathbf{x} - \mathbf{y}' \in P$ for $\mathbf{y}' = \lambda\mathbf{y}$, $\lambda > 0$ which will imply $\mathbf{x}$ is not a vertex. Since $A\mathbf{y} = \mathbf{0}$, we only need to show $\mathbf{x} + \mathbf{y}', \mathbf{x} - \mathbf{y}' \geq \mathbf{0}$. This can be achieved by choosing $\lambda = \min_{j: y_j \neq 0} \left\{ \frac{x_j}{|y_j|} \right\} > 0$. □

In other words, the number of non-zero variables in a solution is at most the number of linearly independent tight constraints. A *basic solution* is characterized by a set of n linearly independent tight constraints. A basic solution determines a vertex but there may be more than one basic solution for a vertex (when there are more than n tight constraints at a vertex).

When analyzing an iterative rounding algorithm, usually we take a basic solution, remove the variables that are zero, and argue that if there is no variable of large value, then there are more variables than linearly independent tight constraints, which contradicts the fact that the solution is basic. In the following we illustrate the ideas for the problem of bipartite matching and generalized Steiner network problem. For minimum bounded degree spanning tree, we show how to obtain a good solution with slight violation on the packing constraint, which can be considered as an extension of the iterative rounding technique.

2.3.2 Bipartite Matching

The iterative rounding technique can be used to show the polytope of the bipartite matching is integral. We will sketch the proof here. Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a weight function $w : E \to \mathbb{R}$, the LP relaxation of the bipartite matching problem can be written as follows:

$$\text{(BM)} \qquad \begin{aligned} \max \quad & \sum_{e \in E} w_e\, x_e \\ \text{s.t.} \quad & x(\delta(v)) \leq 1 \qquad \forall\, v \in V_1 \cup V_2 \\ & 0 \leq x_e \leq 1 \qquad \forall\, e \in E \end{aligned}$$

Call a vertex v *tight* if $x(\delta(v)) = 1$.

Theorem 2.6. *In a basic solution to (BM), there is an edge with* $x_e = 1$.

Proof. In a basic solution of (BM), deleting edges with $x_e = 0$ will not affect the fractional solution. After that, if there is no edge with $x_e = 1$, every edge is fractional. Then the degree of each tight vertex is at least 2. This means the number of edges is at least the number of tight vertices. Since the solution is basic, the number of non-zero variables (edges) is at most the number of tight vertices. Therefore the number of tight vertices equals the number of edges and each tight vertex is of degree exactly 2. Since the graph is bipartite and every vertex is tight, the sum of all degree constraints on vertices of V_1 include every edge once. Similarly, the sum of degree constraints on vertices of V_2 include every edge once. This means the set of tight degree constraints are not linearly independent, and so the solution is not basic since the number of linearly independent constraints is strictly less than the number of variables. Therefore there must be an edge with $x_e = 1$. □

For an edge of value 1, every edge that intersect it would have zero value by the degree constraint. This means in the support of the basic solution, the edge with value 1 is "isolated". Therefore when we pick that edge and remove all the neighbouring edges, the fractional solution remains basic so Theorem 2.6 can be applied repeatedly. This shows the polytope of bipartite matching is integral.

2.3.3 Generalized Steiner Network Problem

Jain applied this technique to approximate the generalized Steiner network problem. In this problem we are given a graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}$ and a requirement function $r : V^2 \to \mathbb{Z}$ on pairs of vertices. The goal is to find a set of edges of minimum cost such that in the graph G' spanned by the edges, there are $r(u, v)$ edge disjoint paths between vertices u and v. By the max-flow-min-cut theorem, this means for every cut $S \subseteq V$, there are $f(S) = \max_{u \in S, v \notin S} r(u, v)$ edges crossing this cut. The problem can be written as the following LP:

$$\text{(GSN)} \qquad \begin{array}{lll} \min & \sum_{e \in E} c_e x_e & \\ \text{s.t.} & x(\delta_G(S)) \geq f(S) & \forall S \subseteq V \\ & 0 \leq x_e \leq 1 & \forall e \in E \end{array}$$

Theorem 2.7 ([24]). *In a basic solution to* (GSN), *there is an edge with* $x_e \geq \frac{1}{2}$.

With Theorem 2.7, Figure 2.9 shows the iterative rounding algorithm for the Generalized Steiner Network problem.

In the algorithm, when we update $f(S)$, the reduced problem is of the same form so Theorem 2.7 still applies. Call the reduced instance (GSN$'$) and let F' be an integral solution to (GSN$'$). Let z_r^* and z^* be the cost of the optimal solution for the (GSN$'$) and (GSN) respectively. The following theorem states that cost of the solution $F = F' \cup H$ is 2-approximate when combining a 2-approximate solution F' for the reduced instance with H.

```
Algorithm GSN
Input: A graph $G$, a cost function $c$ and a requirement function $f$
Output: A set of edges $F$ satisfying the requirement,
        with cost at most twice the optimal
$F \leftarrow \emptyset$;
While $f(S) > 0$ for some $S \subseteq V$ do
    Solve (GSN) and obtain a basic solution $x$;
    Let $H$ be the set of edges with value at least $\frac{1}{2}$;
    $F \leftarrow F \cup H$;
    Update the requirement $f(S)$;
    $G \leftarrow G - H$;
return $F$;
```

Figure 2.9: Iterative rounding algorithm for generalized Steiner network problem

Theorem 2.8. *If F' has a value of at most $2z_r^*$, then $F = F' \cup H$ is an integral solution to (GSN) with value at most $2z^*$.*

Proof. It is clear that F is a feasible solution to (GSN). Note that if we restrict the solution x^* to $G - H$, it is a feasible solution to (GSN′). Hence

$$2z_r^* \leq 2z^* - \sum_{e \in H} x_e^* c_e,$$

i.e.,

$$2z^* \geq 2z_r^* + \sum_{e \in H} x_e^* c_e \geq 2z_r^* + \sum_{e \in H} c_e,$$

where the last inequality is due to the fact that $x_e^* \geq \frac{1}{2}$ when $e \in H$. Therefore $F = F' \cup H$ is a integral solution to (GSN) since F' has a value of a most $2z_r^*$. □

Since we pick an edge only when its values is at least $\frac{1}{2}$, the solution obtained is 2-approximate. Next we show the detail steps for proving the existence of an edge of large value. To illustrate the ideas, it suffices to show a simplified version where we pick the edge when $x_e \geq \frac{1}{3}$.

Consider a basic solution of (GSN). If there is an edge of value 1, we are done; if there is an edge of zero value, we can delete it without affecting the LP solution. After these operation, we assume all edges have fractional value. In this case the basic solution is defined by a *laminar family* of tight sets: for any two tight sets in the family, either they are disjoint or one contains the other.

Let the maximal laminar family be $\mathcal{L}$. Now, if every edge has value $0 < x_e < \frac{1}{3}$, we show that the number of linearly independent constraints $|\mathcal{L}|$ is less than the number

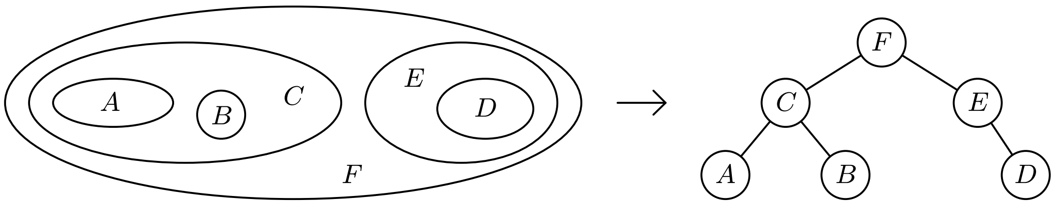


Figure 2.10: Laminar family represented by a forest

of edges (variables), thus obtaining a contradiction. This is done by a token counting argument.

First, assign 2 tokens to each edge uv. The tokens on the edge are redistributed so that both u and v receive one token. The tokens collected by a vertex are given to the smallest tight set in $\mathcal{L}$ that contains it. Since $\mathcal{L}$ is laminar, for any two sets in $\mathcal{L}$, either they are disjoint or one contains the other. We can represent the relationship by a forest: a set A is a descendent of B if $A \subset B$ (Figure 2.10).

We claim that every node in the tree receives 2 tokens and has at least 2 extra tokens to pass to its parent. This would mean that there are extra tokens at the root and contradict the fact that the solution is basic. A leaf node L on the tree collects 4 tokens because $\delta_G(L) = f(L) \geq 1$ but $x_e < \frac{1}{3}$ so there are at least 4 edges that cross this set. L keeps 2 tokens and pass the remaining tokens (≥ 2) to its parent. For a non-leaf node, if it has at least 2 children, it can keep 2 tokens and pass the remaining tokens to its parent; otherwise, it has one child only. Let the node be B and its child be C. There must be edges that cross B but not C for otherwise they are the same cut and are not linearly independent. If there are at least two edges that cross B but not C, we get two extra tokens for B; if there is only one edge e that crosses B but not C, there is at least one edge that goes from C to $B - C$ since otherwise $\delta_G(B) = \delta_G(C) + x_e$ so $\delta_G(B)$, $\delta_G(C)$ cannot be both integers. By establishing the claim, we prove that there is an edge of value at least $\frac{1}{3}$.

As a remark, iterative rounding is currently the only method that yields a constant factor approximation for the Generalized Steiner Network problem.

2.3.4 Minimum Bounded Degree Spanning Tree

Besides covering problems, iterative rounding can also be used for packing problems when we allow a slight violation on the packing constraint. The problem of Minimum Bounded Degree Spanning Tree (MBDST) asks to find a spanning tree of minimum cost while satisfying the degree bound B_v imposed on vertices (which may vary among vertices). By setting $B_v = 2$ for all vertices, this problem asks to find a minimum cost Hamiltonian path, which is NP-hard.

An algorithm for this problem is $+\alpha$-approximate if it returns a spanning tree with cost at most the optimal tree and degree of each vertex is at most $B_v + \alpha$. Assuming P $\neq$ NP, there is no polynomial time approximation algorithm that satisfies all the degree bounds with guarantee on the cost of the tree [17]. Goemans [18] showed a +2-approximation algorithm that finds a spanning tree with degree violation of at most 2. Singh and Lau [29] showed a +1-approximation algorithm.

The LP for MBDST can be written as follows (initially $W = V$):

$$\begin{array}{llrll}
\mathrm{MBDST}(G, \mathcal{B}, W) & \min & \displaystyle\sum_{e \in E} c_e\, x_e & & \\
& \text{s.t.} & x(E(V)) = |V| - 1 & & (1) \\
& & x(E(S)) \leq |S| - 1 & \forall\, S \subset V & (2) \\
& & x(\delta(v)) \leq B_v & \forall\, v \in W & \\
& & x_e \geq 0 & \forall\, e \in E &
\end{array}$$

In this section we show how to find a spanning tree with violation at most 2 [29] using the idea of iterative rounding, which is easier to analyse. The key to a solution with cost no more than the optimal is that we don't round. We pick the edge e only when it is incident to a vertex v of degree 1 (and therefore $x_e = 1$ by constraints (1) and (2) with $S = V - \{e\}$). We need to show that when there is no vertex of degree 1, there is a tight vertex v of degree at most 3. Then we forget the degree constraint on v and solve the LP again. Even if all edges incident to v are picked in the solution we violate the degree constraint of v by at most 2 since $B_v \geq 1$ for any $v \in V$. By relaxing the constraint, the optimal solution can only decrease in cost, therefore obtaining a +2-approximation algorithm.

As in the analysis above, we first delete 0-edges. Then we prove that if there is no vertex of degree 1 (leaf vertex), there exists a tight vertex degree at most 3. The following lemma is a result of the fact that the set of linearly independent tight constraints forms a laminar family.

Lemma 2.9. *There are at most $|V| - 1$ linearly independent tight constraints of type* (1) *and* (2)*.*

Theorem 2.10. *If every edge e in* MBDST$(G, \mathcal{B}, W)$ *has $x_e > 0$ and there is no vertex of degree* 1*, then there is a tight vertex of degree at most* 3*.*

Proof. Assume there is no vertex of degree 1 and there is no tight vertex of degree at most 3. Then each vertex in W has degree at least 4, and each vertex in $V - W$ has degree at least 2. Therefore $|E| \geq \frac{1}{2} \times (4 \cdot |W| + 2 \cdot (n - |W|)) = n + |W|$. By Lemma 2.9, there are at most $n-1$ linearly independent tight constraints of type (1) and (2), we get a contradiction since the total number of linearly independent tight constraints is at most $n-1+|W|$ which is less than the number of edges. □

With Theorem 2.10, the algorithm in Figure 2.11 will give a +2-approximation algorithm.

```
Algorithm MBDST
Input: A graph $G$, degree bound $\mathcal{B} = \{B_v\}$ on vertices of $G$
Output: A set $F$ of edges forming a spanning tree with degree on each vertex bounded
by $B_v + 2$ and cost no more than the optimal spanning tree.
Initialize $F \leftarrow \emptyset$, $W \leftarrow V$;
While $V(G) \neq \emptyset$ do
      Solve MBDST$(G, \mathcal{B}, W)$ to obtain a basic solution $x$;
      Remove all edges $e$ with $x_e = 0$ from $G$;
      If there exists a vertex $v \in V$ with only one edge $e = uv$ incident to it then
            update $F \leftarrow F \cup \{e\}$, $G \leftarrow G - \{v\}$, $W \leftarrow W - \{v\}$, and $B_u \leftarrow B_u - 1$;
      If there exists a vertex $v \in W$ such that $\deg(v) \leq 3$ then
            update $W \leftarrow W - \{v\}$;
return $F$;
```

Figure 2.11: +2-approximation algorithm for the MBDST problem

2.4 Packing Problems

Although the Ryser's conjecture is open for $k \geq 4$, the fractional (linear program) formulation was studied by Füredi [15] and Füredi, Kahn and Seymour [16].

Unweighted Matching

Füredi [15] investigated the unweighted matching problem. In order to outline the proof, we need some notations. The *matching number* ν is the size of the largest matching. The *fractional matching number* ν^* is the maximum value of a fractional matching, i.e. the maximum sum of weights $(0 \leq w(e) \leq 1)$ assigned to edges so that at each vertex, total weight of all edges incident to it is at most 1. It is the objective value of the LP relaxation for the problem. A *transversal*, or vertex cover, is a subset of vertices such that together they meet all edges. The *fractional transversal number* τ^* is minimum value of a fractional transversal, i.e. the minimum sum of weights $(0 \leq t(v) \leq 1)$ assigned to vertices so that for each edge e, $\sum_{v \in e} t(v) \geq 1$.

He proved that if there is no projective plane (see below) in a k-uniform hypergraph, then the fractional matching number is upper bounded by $k-1$ times the matching number. The proof is by induction on the matching number and is non-constructive. By LP duality, the fractional matching number is equal to the fractional transversal number. Therefore it suffices to give a fractional transversal of size at most $(k-1)\nu$.

Property of a basic solution is used so we assume the number of non-zero edges $|E|$ is at most the number of vertices $|V|$. Consequently,

$$(*) \qquad \min_{v\in V} \deg(v) \le \frac{\sum\{\deg(v) : v \in V\}}{|V|} = \frac{k|E|}{|V|} \le k.$$

The proof is divided into two cases. If there is a vertex v of degree $l < k$, let the l edges incident to it be $E_1, E_2, \ldots, E_l$. By removing an edge E_i together with all edges incident to it, we get a smaller graph G_i in which the matching number is decreased by at least one (since any matching in G_i together with E_i is a valid matching in the original graph). From the induction hypothesis, there is a small fractional transversal t_i for G_i. A fractional transversal of the original graph is obtained by combining the t_i's.

If there is no vertex of small degree, then by (*) the graph is k-regular. In this case he proved that the graph is small, so the fractional transversal is bounded trivially.

Weighted Matching

The weighted version of the theorem is proved by Füredi, Kahn and Seymour [16]. Again the proof is non-constructive. They considered a minimal counterexample H. LP duality is used where they show that when the integral weighted matching in H is 1, the fractional weighted matching is upper bounded by $k-(k-1)x(e)$ for arbitrary $e \in H$. Therefore to obtain the desired result (fractional weighted matching is at most $k-1$), it suffices to show an edge e with $x(e) \ge \frac{1}{k-1}$. We focus on basic solution as in the unweighted case.

Assume $x(e) < \frac{1}{k-1}$ for all $e \in H$. Then the graph is k-regular and the number of tight vertices is equal to the number of edges. In this case there is only one optimal fractional matching, namely $x(e) = \frac{1}{k}$ for all $e \in H$. Since every edge is intersected by at most k^2-k other edges, and H does not contain k^2-k+1 pairwise intersecting edges (for otherwise they form a projective plane), Brooks' theorem[2] (see [9]) implies that the graph H can be decomposed into k^2-k matchings M_i, $1 \le i \le k^2-k$. Therefore

$$\sum_{e\in H} w(e)x(e) = \frac{1}{k}\sum_{e\in H} w(e) = \frac{1}{k}\sum_{i=1}^{k^2-k}\sum_{e\in M_i} w(e) \le \frac{1}{k}(k^2-k).$$

[2]In short, Brooks' theorem states that in a general graph with maximum degree Δ that is neither a clique nor an odd cycle, the vertices may be coloured with only Δ colours.

2.4.1 Projective Plane

Projective plane and truncated projective plane are used in showing that the integrality gap analyses are tight.

A *finite projective plane* (projective plane for short) of order $n \geq 2$ is a combinatorial object formed by *lines* and *points* such that the following properties are satisfied:

1. Any two lines intersect at one point,
2. Any two points determines one line,
3. Every line contains $n+1$ points, and
4. Every point has $n+1$ lines on it.

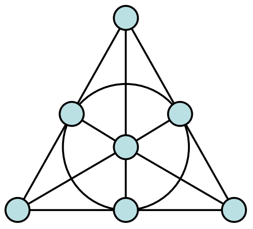


Figure 2.12: Projective plane of order 2

Property 1 states that there are no parallel lines in a projective plane; properties 3 and 4 together implies that a projective plane of order n contains n^2+n+1 points and lines. If we take the dual of the projective plane by swapping the lines and points, the result is still a projective plane.

As an example, a projective plane of order 2 is shown in Figure 2.12 (also known as a Fano plane). There are 7 lines and 7 points. Note that a “line” is not necessarily a straight line; it can be an arbitrary line to show the relationship of the points.

Not all orders for the projective planes are possible. The Bruck-Ryser-Chowler theorem states that if the order n is congruent to 1 or 2 (mod 4), then it must be the sum of two squares. This rules out $n = 6, 14, 15, \ldots$ On the other hand, there is a projective plane of order n if n is a prime power. The case $n = 10$ is ruled out by massive computer search [26] (so the Bruck-Ryser-Chowler theorem is necessary but not sufficient). Nothing more is known; in particular $n = 12$ is open (to see why it is difficult to search for a solution, observe that this problem is equivalent to filling an $(n^2+n+1) \times (n^2+n+1)$ matrix by 0 and 1 so that each column and each row has exactly $(n+1)$ 1’s and every two rows (columns) has inner product exactly 1).

When we treat the lines in a projective plane as hyperedges, a projective plane of order n is an $(n+1)$-uniform hypergraph and it is an intersecting hypergraph with $n+1$ edges. A projective plane of order $k-1$ (if it exists) serves as an example showing the integrality gap of the standard relaxation of k-Set Packing is at exactly $k-1+\frac{1}{k}$: if we set $x_e = \frac{1}{k}$ for all edges in the projective plane, all degree constraints are satisfied because it is a k-regular graph. The fractional solution is $(k^2-k+1)\left(\frac{1}{k}\right) = k-1+\frac{1}{k}$, while the integral solution is 1 because the graph is intersecting. The integrality gap cannot be

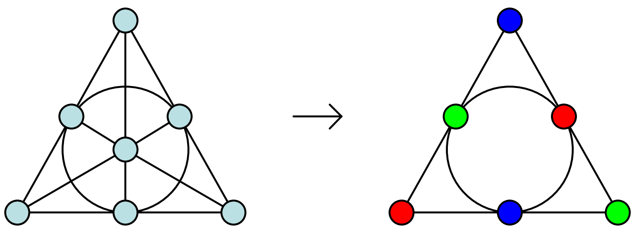


Figure 2.13: Removing the point in the center, we get a truncated projective plane of order 2, which is 3-partite.

bigger since all degree constraints are tight in this assignment. Therefore the integrality gap analysis in Section 4.1 is tight when $k-1$ is a prime power.

Truncated Projective Plane

If we remove one of the points and all lines that contain it, we obtain a *truncated projective plane*. A truncated projective plane of order n contains n^2+n points and n^2 lines. It is an $(n+1)$-partite hypergraph, which can be seen by colouring the points by $n+1$ colours so that each of the remaining lines has all the colours: let the lines incident to a particular point p_0 (to be removed) in the projective plane be $L_i = (p_0, p_i^1, p_i^2, \ldots, p_i^n)$ for $i = 1, 2, \ldots, n+1$. Colour the points on line L_i (except p_0) by colour i (see Figure 2.13). It is easy to check that points on any of the remaining n^2 lines are coloured by a different colour since every two lines in a projective plane intersect at exactly one point. Therefore the graph obtained is $(n+1)$-partite. Also, it is also easy to check that each of the points (not including p_0) is of degree n, i.e. a truncated projective plane of order n is both n-regular and $(n+1)$-partite.

A truncated projective plane of order $k-1$ (if it exists) serves as an example showing that the integrality gap of the standard relaxation of k-D Matching is at least $k-1$: this time we set $x_e = \frac{1}{k-1}$ because the graph is $(k-1)$-regular. The fractional solution is $(k-1)^2\left(\frac{1}{k-1}\right) = k-1$ while the integral solution is 1 as the graph is intersecting.

Construction of Projective Plane of Prime Order

Here we present a way to construct a projective plane of prime order. Note that this method does not apply to projective plane of prime power order.

Let n be a prime. To construct a projective plane of order n, label the (n^2+n+1) points as P, $P(c)$ and $P(r,c)$, where $r, c = 0, 1, \ldots, n-1$. Then there are 3 types of lines:

- 1 line $L = \{P, P(0), P(1), \ldots, P(n-1)\}$.
- n lines $L(c) = \{P, P(0,c), P(1,c), \ldots, P(n-1,c)\}$, where $c = 0, 1, \ldots, n-1$.

- n^2 lines $L(r, c) = \{P(c), P((r + ci) \bmod n, i)\}$, where $i, r, c = 0, 1, \ldots, n - 1$.

The expression $((r + ci) \bmod n)$ will go through each value as i varies from 0 to $n - 1$, but only if n is prime. For prime power n, the projective plane can be constructed using homogeneous coordinates over a finite field.

2.5 Local Ratio

Local ratio is an algorithmic framework used for design and analysis of algorithms for optimization problems. It is simple yet can be applied to a broad range of problems. A comprehensive survey on this technique can be found in [5].

We first take a look at a simple example of vertex cover before we go into the local ratio theorem.

2.5.1 Vertex Cover

In vertex cover, each vertex is associated with a weight and the goal is to select a subset of vertices such that each edge is incident to at least one of them and the total weight of selected vertices is minimized. A 2-approximation algorithm for vertex cover can be obtained as follows. Think of the weight $w(v)$ of a vertex v as the price of "buying" it. Instead of buying a vertex in one payment, we make several down payments so that we actually buy the vertex if in the end the sum of down payments is equal to the price of the vertex. When we pay a down payment of $\$\epsilon$ on a vertex v, the price tag of v is decreased by $\$\epsilon$. Therefore we have a vertex cover when every edge is incident to at least one zero-cost vertex.

Whenever there is an edge (u, v) such that $\min\{w(u), w(v)\} > 0$, we pay a down payment of $\min\{w(u), w(v)\}$ to both u and v. The price of u and v are decreased by $\min\{w(u), w(v)\}$ as a result. This procedure will make the price of either vertex to be zero so the process will terminate in at most $|V|$ iterations. Now we bound ratio of the payments we made and the optimal solution.

Let (u_i, v_i) be the edge selected in the i-th round and ϵ_i be the down payment made on each of the endpoints. We pay $2\epsilon_i$ and the price of each of u_i and v_i is decreased by ϵ_i so the total price of the optimal vertex cover is decreased by at least ϵ_i (because any solution must take either u_i or v_i, or both). Therefore the *local ratio* between our payments and the drop in optimal solution is 2. Sum over all down payments, our solution, which is the set of all "free" vertices, is a 2-approximate solution.

2.5.2 Local Ratio Theorem

The problem we are considering is minimization problem: given a weight vector $w \in \mathbb{R}^n$ and a set of constraints $\mathcal{C}$, find a solution $x \in \mathbb{R}^n$ satisfying the constraints in $\mathcal{C}$ and that the scalar product $w \cdot x$ is minimized. In the vertex cover problem above, w stands for the vector of weights on vertices and x is the characteristic vector of the solution, such that $x(v) = 1$ if vertex v is in the vertex cover and $x(v) = 0$ otherwise.

Theorem 2.11 (Local ratio)**.** *Let $\mathcal{C}$ be a set of constraints on vectors in $\mathbb{R}^n$. Let $w, w_1, w_2 \in \mathbb{R}^n$ be weight vectors such that $w = w_1 + w_2$. Let $x \in \mathbb{R}^n$ be a feasible solution (with respect to $\mathcal{C}$) so that it is r-approximate with respect to both w_1 and w_2. Then x is r-approximate with respect to w.*

Proof. Let x^*, x_1^* and x_2^* be optimal solutions with respect to w, w_1 and w_2 respectively. By the optimality of x_1^* and x_2^*, We have $w_1 \cdot x_1^* \le w_1 \cdot x^*$ and $w_2 \cdot x_2^* \le w_2 \cdot x^*$. Therefore

$$\begin{aligned} w \cdot x &= w_1 \cdot x + w_2 \cdot x \\ &\le r(w_1 \cdot x_1^*) + r(w_2 \cdot x_2^*) \\ &\le r(w_1 \cdot x^*) + r(w_2 \cdot x^*) \\ &= r(w \cdot x^*). \end{aligned}$$ □

With the local ratio theorem, we can obtain approximation algorithms if we can split the problem and weight function into two parts and prove that the solution is r-approximate with respect to both w_1 and w_2. Usually, by the use of induction, we can assume that the solution to the subproblem with weight vector w_2 is r-approximate, so we can focus on the design of w_1 and show that the solution is r-approximate with respect to w_1.

2.5.3 Feedback Vertex Set in Tournaments

A *tournament* is an orientation of edges in a complete graph, that is, for each pair (u, v) of nodes, either we have the arc (u, v) or (v, u) but not both. The feedback vertex set in tournament problem is that, given a tournament and a weight function on its nodes, remove a subset of nodes of minimum total weight such that the remaining graph does not contain a directed cycle. We now present a 3-approximation algorithm using local ratio. First we have the following lemma which states that we only need to take care of directed triangle (a cycle of length 3):

```
Algorithm FVST($G, \mathbf{w}$)
Input: A tournament $G$, a weight vector $\mathbf{w}$ on the set of nodes
Output: A set of nodes $S$ to remove such that the resulting graph
        does not contain a directed cycle
If there is a positive triangle then
    Let the positive triangle be $\{v_1, v_2, v_3\}$;
    Let $\epsilon = \min\{w(v_1), w(v_2), w(v_3)\}$;
    Decompose the weight vector $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$ where
        $w_1(v) = \begin{cases} \epsilon & \text{if } v \in \{v_1, v_2, v_3\}, \\ 0 & \text{otherwise.} \end{cases}$
    return FVST($G, \mathbf{w}_2$);
else
    return the set of zero weight nodes in $G$.
```

Figure 2.14: Algorithm for feedback vertex set in tournament

Lemma 2.12. *A tournament contains a directed cycle if and only if it contains a directed triangle.*

Proof. Let the minimum length of a directed cycle be $k \geq 3$ (it cannot be 2 since either arc (u, v) or arc (v, u) exists but not both). If $k = 3$ we are done. Otherwise let the directed cycle be $v_1, v_2, \ldots, v_k, v_1$ where v_i points to v_{i+1}, $1 \leq i < k$ and v_k points to v_1. If the arc (v_3, v_1) exists, then we have a directed triangle; otherwise there is a directed cycle $v_1, v_3, v_4, \ldots, v_k, v_1$ of length $k-1$, which violates our assumption that the minimum length of a directed cycle is k. □

Call a directed triangle *positive* if all three nodes have strictly positive weights. The algorithm is shown in Figure 2.14.

We prove that the algorithm returns a 3-approximate solution by induction on the number of nodes having strictly positive weight. At the base case, every node is of zero weight so our solution is 3-approximate. In the inductive step, when there is a positive triangle, at least one of the nodes has to be removed. For $\mathbf{w}_1$, we pay at most 3ϵ while the optimal solution is at least ϵ, therefore the solution is 3-approximate with respect to $\mathbf{w}_1$. By the way we define w_1, number of strictly positive nodes in w_2 is less than that of w, therefore the solution is 3-approximate with respect to w_2. By Theorem 2.11, the solution is 3-approximate with respect to w. We remark that there is a 2.5-approximation algorithm due to Cai, Deng and Zang [10], by considering the structure of forbidden subtournament.

2.5.4 Fractional Local Ratio

In the examples above, at each level of recursion, we compare the solution found to the optimal solution, with respect to w_1 and w_2. There may be two different optimas corresponding to the two weight functions. Also, we assume the solution x to be integral. In fractional local ratio, there is only one optimal x^* for comparison, usually comes from the solution of the LP relaxation. x^* serves as a lower bound to the problem, and may not be feasible to the original problem. Nevertheless, Theorem 2.11 can be used in this paradigm without much modification. In the following we will see an example of how this idea is applied to the problem of maximum weight independent set in t-interval graph [6]. Our algorithm for k-D Matching is inspired by this work.

2.5.5 Maximum Weight Independent Set in t-interval Graph

A t-interval system is a collection $\{I_1, I_2, \ldots, I_n\}$ of sets, where each I_i consists of up to t disjoint intervals. A t-interval graph is the intersection graph of a t-interval system: each vertex v_i corresponds to a set I_i, and there is an edge (v_i, v_j) if some interval in I_i intersect some interval in I_j. If an interval $[a, b]$ is shared by two vertices, we make copies of the interval so that each interval belong to only one vertex.

Given a point p and a vertex v, we denote by $p \in\in v$ the statement that the point p is contained in some interval of v. Let R be the set of right endpoints of intervals in the system. Denote by $N[v]$ the set $N(v) \cup \{v\}$. The algorithm begin by solving the following LP relaxation to obtain an optimal fractional solution x^*:

$$\begin{array}{lll} \max & \sum_{u \in V} w(u)\, x_u & \\ \text{s.t.} & \sum_{u : p \in\in u} x_u \leq 1 & \forall\, p \in R \\ & 0 \leq x_u \leq 1 & \forall\, u \in V \end{array}$$

With the optimal fractional solution x^*, Figure 2.15 shows a $2t$-approximation algorithm [6] (i.e. $w \cdot x \geq \frac{1}{2t} w \cdot x^*$). First we claim that $S \subseteq V_+$. At the base case ($E_+ = \emptyset$), $S = V_+$ and $w \cdot x = \sum_{u:w(u)>0} w(u) \cdot 1 \geq \sum_{u:w(u)>0} w(u) \cdot x^*_u \geq \sum_{u \in V} w(u) \cdot x^*_u = w \cdot x^*$. Since $w \cdot x \geq 0$, $w \cdot x \geq \frac{1}{2t} w \cdot x^*$. In the inductive step, let x' be the incidence vector of S' and $V'_+ = \{u : w_2(u) > 0\}$. By the inductive hypothesis and the definition of w_2, $S' \subseteq V'_+ \subseteq V_+$. Since $v \in V_+$, we have $S \subseteq V_+$. By the inductive hypothesis, $w_2 \cdot x' \geq \frac{1}{2t} w_2 \cdot x^*$. Note that $w_2(v) = 0$, so $w_2 \cdot x = w_2 \cdot x'$, therefore x is $2t$-approximate with respect to w_2. The *if* statement at the end of the algorithm ensures that at least one vertex from $N[v]$ will appear in our solution, so $w_1 \cdot x \geq \epsilon$. To finish the proof, we need to

```
Algorithm t-interval($G, w, x^*$)
Let $V_+ = \{u \mid w(u) > 0\}$ and $G_+ = (V_+, E_+)$ be the subgraph of $G$ induced by $V_+$;
**If** $E_+ = \emptyset$ **return** $V_+$;
Let $v \in V_+$ be a vertex minimizing $\sum_{u \in N[v] \cap V_+} x^*_u$ and let $\epsilon = w(v)$;
Define the weight functions
    $$w_1(u) = \begin{cases} \epsilon & \text{if } u \in N[v] \cap V_+, \\ 0 & \text{otherwise,} \end{cases}$$
and $w_2 = w - w_1$;
$S' \leftarrow$ t-interval($G, w_2, x^*$);
**If** $S' \cup \{v\}$ is an independent set **then**
    **return** $S = S' \cup \{v\}$;
**else**
    **return** $S = S'$;
```

Figure 2.15: Algorithm for maximum weight independent set in t-interval graph

bound $w_1 \cdot x^*$. The lemma below implies this and shows the solution x is $2t$-approximate, by the fractional local ratio theorem.

Lemma 2.13. *The vertex v minimizing $\sum_{u \in N[v] \cap V_+} x^*_u$ satisfies $\sum_{u \in N[v] \cap V_+} x^*_u \leq 2t$.*

Proof. We prove this lemma by showing that there is a vertex s such that $\sum_{u \in N[s] \cap V_+} x^*_u \leq 2t$. This condition is satisfied if there is any isolated vertex. Therefore assume G_+ contains no isolated vertex. An equivalent claim would be $x^*_s \sum_{u \in N[s] \cap V_+} x^*_u \leq 2t x^*_s$. Sum over all s, we can prove this lemma by showing that $\sum_{s \in V_+} x^*_s \sum_{u \in N[s] \cap V_+} x^*_u \leq 2t \sum_{s \in V_+} x^*_s$.

Let $I \sim J$ denote the statement that intervals I and J intersect and $J \sim_R I$ denote J contains the right endpoint of I. If interval I belongs to vertex v, define $x^*_I := x^*_v$. Then,

$$\begin{aligned} \sum_{s \in V_+} x^*_s \sum_{u \in N[s] \cap V_+} x^*_u &\leq \sum_{I \sim J} x^*_I x^*_J \\ &\leq 2 \sum_{J \sim_R I} x^*_I x^*_J \\ &= 2 \sum_{s \in V_+} \sum_{I \in s} \sum_{J \sim_R I} x^*_I x^*_J \\ &= 2 \sum_{s \in V_+} x^*_s \sum_{I \in s} \sum_{J \sim_R I} x^*_J. \end{aligned}$$

The independent set constraint states that $\sum_{J \sim_R I} x^*_J \leq 1$, and since there are at most t intervals in any vertex s, $\sum_{I \in s} \sum_{J \sim_R I} x^*_J \leq t$. □

Chapter 3

k-Dimensional Matching

The results in this chapter are based on joint work with Lap Chi Lau [12].

3.1 Integrality Gap of the Standard LP Relaxation

In this section we prove Theorem 1.1, namely the integrality gap of the standard LP relaxation of *weighted* k-Dimensional Matching is $k-1$ for $k \geq 3$, and show that the analysis is tight in general. We look at a basic solution of the LP and argue that there is a vertex of small degree. Using this property, we can pick an edge incident to it without losing too much for the fractional solution. By fractionally colouring the solution, we prove that the integrality gap is $k-1$. The truncated projective plane in Section 2.4.1 shows that the integrality gap analysis is tight.

With the shorthand $x(S)$ for $\sum_{e \in S} x_e$ and $\delta(v)$ denotes the set of edges incident to a vertex v, the standard LP relaxation for the problem can be written as:

$$\text{(LP)} \qquad \begin{aligned} \max \quad & \sum_{e \in E} w_e\, x_e & \\ \text{s.t.} \quad & x(\delta(v)) \leq 1 & \forall\, v \in V \\ & x_e \geq 0 & \forall\, e \in E \end{aligned}$$

Consider a solution to (LP). If there is an edge e with $x_e = 0$, delete it. After all the zero edges are removed, consider a basic solution of (LP). Denote by T the set of constraints that are tight, and E' the set of non-zero edges. In a basic solution, we have $|E'| \leq |T|$. Next we argue that there is a vertex of small degree in the basic solution.

Proposition 3.1. *In a basic solution of* (LP)*, there is a vertex* $v \in T$ *with* $\deg(v) \leq k-1$.

Proof. Suppose every vertex in T has degree at least k. We have $\sum_{v \in T} \deg(v) \geq k \cdot |T|$. On the other hand, every edge consists of k vertices, so $\sum_{v \in V} \deg(v) = k \cdot |E'| \geq$

$\sum_{v \in T} \deg(v) \geq k \cdot |T|$. Together with the fact that $|E'| \leq |T|$ in a basic solution, we have $|E'| = |T|$ and each vertex in T is covered by exactly k edges. This means every edge in E' must consist of vertices in T only. Since the graph induced by edges of E' is k-regular and k-partite, the sum of all constraints incident to the first partition includes every edge once. Similarly, the sum of all constraints incident to the second partition includes every edge once. This shows that the set of constraints are not linearly independent, and hence the solution cannot be basic since the number of linearly independent constraints is less than the number of variables. □

Remark 1. *The above proof also holds when the graph is bipartite k-uniform hypergraph. In this case, the vertex set V is composed of 2 disjoint set of vertices X and Y, and every hyperedge has one vertex in X and the remaining $k-1$ vertices in Y. Sum of all constraints incident to X includes every edge once and sum of all constraints incident to Y includes every edge $k-1$ times. Therefore the set of constraints is linearly dependent.*

Remark 2. *When the graph is a general k-uniform hypergraph, the above proposition does not hold, even when there is no projective plane. One example (for $k = 3$) is shown in* Figure 3.1. *When edge weight is uniform, the basic solution to* (LP) *is $x_e = \frac{1}{3}$ for all $e \in E$. This means all vertices have degree* 3.

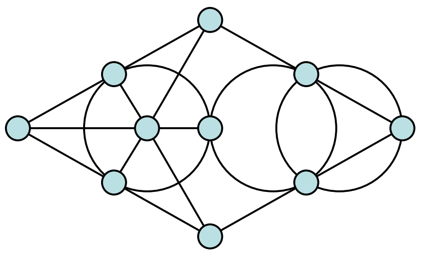


Figure 3.1: A 3-uniform hypergraph that violates Proposition 3.1

3.1.1 Approximation Algorithm for Unweighted k-D Matching

With Proposition 3.1, we obtain an approximation algorithm for the unweighted matching problem ($w_e = 1$ for all edge $e \in E$) using iterative rounding as shown in Figure 3.2.

In Step 3(a) of the algorithm, the existence of a small degree tight vertex is guaranteed by Proposition 3.1. The value of the largest edge is at least $\frac{1}{k-1}$ which means the sum of edge value (excluding e) at each vertex of e is at most $\frac{k-2}{k-1}$. Therefore when we remove all edges incident to e, the fractional solution is decreased by at most $k \cdot \frac{k-2}{k-1} + \frac{1}{k-1} = k - 1$ while we get an integral solution of 1. Hence it is a $(k-1)$-approximation algorithm.

Unweighted k-Dimensional Matching Algorithm

1. Solve (LP) to get a basic solution $\mathbf{x}$.
2. Remove edges with value $x_e = 0$.
3. While there are edges remaining, do
   (a) Select a vertex v with degree at most $k-1$.
   (b) Among all edges incident to v, pick the edge e with the largest value and remove all edges that intersect e.
4. Return the set of picked edges as the solution.

Figure 3.2: $(k-1)$-approximation algorithm for unweighted k-D Matching

Why this does not work for weighted case

Iterative rounding seems to work well for covering problem but it is not easy to extend it to packing problems. In packing problems, we are not allowed to violate the packing (degree) constraint. Therefore the technique for MBDST (Section 2.3.4) cannot be used. Also, for the weighted packing problem, we cannot round up variables with large fractional value as in the unweighted case because it is not clear how to bound the ratio of weight of the picked edge e to the total weight of the close neighbourhood $N[e]$. As an example, in Figure 3.3, if we pick the middle edge with $x_e = 0.8$ to get a weight 2 solution, we possibly lose up to 91 since all the sets in the neighbourhood cannot be chosen.

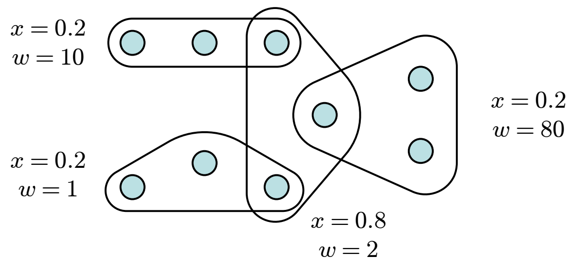


Figure 3.3: Picking sets of large value can lead to a big loss in packing problems

3.1.2 Fractional Colouring

For a set S of edges in E, let χ_S be the characteristic vector of S in $\mathbb{R}^{|E|}$. Let w be the weight vector. The following claim show that if we can express the graph by a convex combination of integral matchings with small sum of coefficients, then we have a good approximate solution.

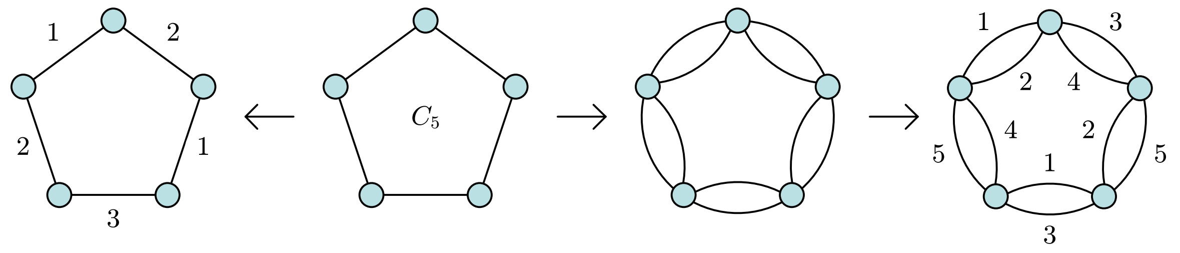


Figure 3.4: The pentagon C_5 is 3 colourable and fractionally 2.5 colourable. (Here $x(e) = 1$ for all e and $L = 2$.)

Claim 3.2. *Let M_i, $i = 1, 2, \ldots, h$ be matchings in the hypergraph G. If there exist coefficients $\lambda_1, \lambda_2, \ldots, \lambda_h \geq 0$ such that $\chi_E = \sum_{i=1}^{h} \lambda_i \chi_{M_i}$ and $\sum_{i=1}^{h} \lambda_i \leq \ell$, then there exists a matching M_j such that it is an ℓ-approximate solution to the hypergraph matching problem.*

Proof. The proof is by an averaging argument. Since $\chi_E = \sum_{i=1}^{h} \lambda_i \chi_{M_i}$, we have $\chi_E \cdot w = \sum_{i=1}^{h} \lambda_i \chi_{M_i} \cdot w$. Suppose $\chi_{M_i} \cdot w < \frac{1}{\ell} \chi_E \cdot w$ for all matchings M_i. Then,

$$\begin{aligned}\sum_{i=1}^{h} \lambda_i \chi_{M_i} \cdot w &< \frac{1}{\ell} \sum_{i=1}^{h} \lambda_i \chi_E \cdot w \\ &= \chi_E \cdot w,\end{aligned}$$

which is a contradiction. Therefore there exists a matching M_j such that $\chi_{M_j} \cdot w \geq \frac{1}{\ell} \chi_E \cdot w$, which is an ℓ-approximate solution to the hypergraph matching problem. □

By showing a convex combination of matchings with a small sum of coefficients, we can show that the integrality gap is small. The matchings in the above claim is related to fractional colouring as follows. In an integral (edge) colouring, each edge is given a colour, so that edges that share the same vertex is assigned different colours. In *fractional colouring*, each edge e is given $x(e)$ colours (which can be fractional). We can interpret fractional colouring as integral colouring in the following sense: pick a large integer L such that $L \cdot x(e)$ is integer for all $e \in E$. Then multiply the graph L times. For example, if $x(e) = 0.3$ and $L = 10$, then we have 3 copies of e in the resulting graph (note that the resulting graph is a multigraph, which contains parallel edges). Colour the edges in the usual way. If N colours are used, then we say the original graph is fractionally N/L colourable (Figure 3.4).

Each of the N colours represent a matching and the original graph is a convex combination of these N matching by taking $\lambda_i = 1/L$. By Claim 3.2, the integrality gap is at most N/L.

To obtain a fractional colouring, it helps to have a good ordering. Suppose we have an ordering of edges $\{e_1, e_2, \ldots, e_m\}$ such that the total fractional value of edges in $N[e_i] \cap$

$\{e_i, e_{i+1}, \ldots, e_m\}$ is at most ℓ. Take a large integer L such that Lx_i is an integer for all i. Make Lx_i copies for each edge e_i and call them $e_{i1}, \ldots, e_{i(Lx_i)}$. Now consider the intersection graph of these hyperedges e_{ij} and we colour them using ℓL colours in the reverse order, i.e. $e_{m(Lx_m)}$ is coloured first and e_{11} is coloured last. Since the value of edges in $N[e_i] \cap \{e_i, e_{i+1}, \ldots, e_m\}$ is at most ℓ, when we consider the edge e_{ij}, number of coloured neighbours in the intersection graph is bounded by $\ell L - 1$, so we can colour it using the remaining colours. This means we can colour this graph by ℓL colours. Therefore the original graph is fractionally ℓ colourable and the integrality gap is ℓ.

3.1.3 Produce an Ordering

Now we show that it is possible to construct the good ordering described in the last section. To simplify our analysis, we consider a more general version of (LP):

$$\text{LP}(G, \mathcal{B}) \qquad \begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & x(\delta(v)) \leq B_v \qquad \forall\, v \in V \\ & x_e \geq 0 \qquad \forall\, e \in E \end{aligned}$$

where $\mathcal{B}$ denotes the vector of all degree bounds $0 \leq B_v \leq 1$ for each vertex $v \in V$. Initially $B_v = 1$ for all $v \in V$. Note that the general version $\text{LP}(G, \mathcal{B})$ is used only in analysis and will not appear in the approximation algorithm below.

Using Proposition 3.1, we can define an ordering of edges as follows. Consider a vertex $v \in T$ with $\deg(v) \leq k - 1$. Pick the edge e incident to it with the largest value x_e and add it to the pool of solutions. Decrease the degree bound B_u by x_e for all $u \in e$. Remove the edge from the graph. Repeat this process until every edge has zero value.

We claim that the solution remains basic for the new LP after we remove the edge and adjust the degree bound, so we only need to solve the LP once. Suppose the remaining solution is not basic, then there is a non-zero $\epsilon \in \mathbb{R}^m$ such that both $x + \epsilon$ and $x - \epsilon$ are in the solution polytope. This means the $\pm\epsilon$ of the original solution are in the solution polytope and contradicts the fact that the original solution is basic.

Denote by P the pool of solutions selected from the above procedure. For an edge e, let $N[e]$ be the set of edges incident to e, including e itself. The following claim states that the total value in the close neighbourhood of an edge e is small ($\leq k - 1$) in the ordering.

Claim 3.3. *Every edge e, when it is picked, satisfies $\sum_{e' \in N[e]} x_{e'} \leq k - 1$.*

Proof. Let the vertices of e be $v_1, v_2, \ldots, v_k$. Without loss of generality let v_1 be the vertex of degree less than k when e is picked. The value associated with e is at least $B_{v_1}/(k-1)$, and the degree bound B_v for other vertices are decreased by at least $B_{v_1}/(k-1)$. This means on the whole, edges of $N[e]$ can be chosen at most

$$B_{v_1} + \sum_{i=2}^{k} \left(B_{v_i} - \frac{B_{v_1}}{k-1} \right) = \sum_{i=2}^{k} B_{v_i} \leq k-1$$

times. □

By the greedy colouring argument in the last section, we conclude that the integrality gap of the standard LP relaxation of the k-Dimensional Matching is $k-1$.

3.2 Approximation Algorithm for Weighted k-D Matching

The fractional colouring argument in the last section does not directly lead to an algorithm that runs in polynomial time because the L in the argument could be large and may not bounded by a polynomial in n. To obtain an efficient approximation algorithm with approximation ratio matching the integrality gap, Local Ratio is used. In this section, we show how to use Local Ratio to obtain a $(k-1)$-approximation for the weighted k-Dimensional Matching. The algorithm first finds a good ordering of edges as in the previous section, then applies the Local Ratio method to obtain a matching that is at least $\frac{1}{k-1}$ of the optimal.

Weighted k-Dimensional Matching Algorithm

```
1. Solve LP($G, \mathcal{B}$) to get a basic solution $\mathbf{x}$ where $B_v = 1$ for all $v$.
2. Initialize $F \leftarrow \emptyset$.
3. For $i$ from 1 to $|E(G)|$ do
    (a) Find an edge $e$ with $x(N[e]) \leq k-1$.
    (b) Set $e_i \leftarrow e$ and $F \leftarrow F \cup \{e_i\}$.
    (c) Remove $e$ from $G$.
4. $M \leftarrow$ Local-Ratio($F, \mathbf{w}$), where $\mathbf{w}$ is the weight vector of the edges.
5. Return $M$.
```

Figure 3.5: $(k-1)$-approximation algorithm for weighted k-D Matching

```
Local-Ratio($F, \mathbf{w}$)

1. Remove from $F$ all edges with non-positive weights.

2. If $F = \emptyset$, then return $\emptyset$.

3. Choose from $F$ the edge $e$ with the smallest index. Decompose the weight vector
   $\mathbf{w} = \mathbf{w}_1 + \mathbf{w}_2$ where

   $w_1(e') = \begin{cases} w(e) & \text{if } e' \in N[e], \\ 0 & \text{otherwise.} \end{cases}$

4. $M' \leftarrow$ Local-Ratio($F, \mathbf{w}_2$).

5. If $M' \cup \{e\}$ is a matching, return $M' \cup \{e\}$; else return $M'$.
```

Figure 3.6: The local ratio routine

After we get an ordering of edges in Step 3 of the algorithm, we can call the **Local-Ratio** subroutine to obtain a $(k-1)$-approximate solution to the weighted k-D Matching problem. It is easy to see that the solution returned by the local ratio routine is a matching, therefore we only need to prove that the cost of the matching is at least $\frac{1}{k-1}$ of the optimum.

Theorem 3.4. *Let* $\mathbf{x}$ *be an optimal fractional solution to* LP$(G, \mathcal{B})$*. The matching* M *returned by the algorithm in* Figure 3.6 *satisfies* $w(M) \geq \frac{1}{k-1} \cdot \mathbf{w} \cdot \mathbf{x}$.

Proof. The proof is by induction on the number of edges having positive weights. The theorem holds in the base case when there are no edges with positive weights. Let e be the hyperedge e chosen in Step 3 of the algorithm. Since e has the smallest index in the ordering, by Claim 3.3, we have $x(N[e]) \leq k-1$. Let $\mathbf{w}, \mathbf{w}_1, \mathbf{w}_2$ be the weight vectors computed in Step 3 of the algorithm. Let $\mathbf{y}'$ and $\mathbf{y}$ be the characteristic vectors for M' and M obtained in Step 4 and Step 5 respectively. Since $w(e) > 0$ and $w_2(e) = 0$, $\mathbf{w}_2$ has fewer edges with positive weights than $\mathbf{w}$. By the induction hypothesis, $\mathbf{w}_2 \cdot \mathbf{y}' \geq \frac{1}{k-1} \cdot \mathbf{w}_2 \cdot \mathbf{x}$. Since $w_2(e) = 0$, this implies that $\mathbf{w}_2 \cdot \mathbf{y} \geq \frac{1}{k-1} \cdot \mathbf{w}_2 \cdot \mathbf{x}$. By Step 5 of the algorithm, at least one edge in $N[e]$ is in M. Since $x(N[e]) \leq k-1$ and $w_1(e') = w(e)$ for all $e' \in N[e]$, it follows that $\mathbf{w}_1 \cdot \mathbf{y} \geq \frac{1}{k-1} \cdot \mathbf{w}_1 \cdot \mathbf{x}$. Therefore, by Theorem 2.11, we have $\mathbf{w} \cdot \mathbf{y} \geq \frac{1}{k-1} \cdot \mathbf{w} \cdot \mathbf{x}$. This shows M is a $(k-1)$-approximate solution to the k-dimensional matching problem. □

Chapter 4

k-Set Packing

The results in this chapter are based on joint work with Lap Chi Lau [12].

In the last chapter, we showed that the integrality gap of the standard LP relaxation for the k-D Matching is $k-1$ and that it is tight. In this chapter we first show that the previous argument can be used to show the integrality gap of the same LP relaxation gives an integrality gap of $k-1+\frac{1}{k}$. Then for the case of matching on 3-uniform hypergraphs, or 3-Set Packing (3-SP), we prove that by adding a local constraint which we call the Fano plane constraint, the integrality gap can be improved from $\frac{7}{3}$ to 2.

4.1 Integrality Gap of the Standard LP Relaxation

To show the integrality gap of the k-Set Packing is $k-1+\frac{1}{k}$, we use a similar strategy as in Proposition 3.1, but this time we show that there is a vertex of degree at most k instead of $k-1$ (recall that T is the set of tight constraints and E' is the set of non-zero edges):

Proposition 4.1. *For the k-Set Packing problem, in a basic solution of* $\mathrm{LP}(G,\mathcal{B})$*, there is a vertex $v \in T$ with* $\deg(v) \leq k$.

Proof. Suppose every vertex in T has degree at least $k+1$, then $\sum_{v\in T}\deg(v) \geq (k+1)\cdot|T|$. We immediately get a contradiction since $k\cdot|E'| = \sum_{v\in V}\deg(v) \geq \sum_{v\in T}\deg(v) > k\cdot|T|$, which means $|E'| > |T|$ and the solution is not basic. $\square$

Proposition 4.1 implies that the constant in Claim 3.3 is to be changed to $k-1+\frac{1}{k}$, by replacing the denominator $(k-1)$ in the proof by k. Then the same analysis will show the integrality gap is $k-1+\frac{1}{k}$ and it is not repeated here. Again, the integrality gap analysis is tight in general, as shown in Section 2.4.1.

4.2 Improved LP Relaxation for 3-SP

In this section, we prove that by adding additional constraints for the Fano planes to (LP), we can improve the integrality gap for the hypergraph matching problem in 3-uniform hypergraphs from $\frac{7}{3}$ to 2. For every seven hyperedges that form a Fano plane P, the *Fano plane constraint* states that the sum of fractional values in this seven edges must not exceed two[1]:

$$x(P) \leq 2 \qquad \forall \text{ Fano plane } P.$$

We call the resulting linear program the Fano linear program, denoted by *Fano-LP*.

Theorem 1.4. *The* Fano-LP *for unweighted* 3*-uniform hypergraphs has integrality gap exactly* 2*.*

The proof of Theorem 1.4 consists of several steps. We characterize the counterexample H to the theorem with the minimum number of hyperedges. First we prove that each edge e in H satisfies $0 < x_e < \frac{1}{2}$ in the basic solution to Fano-LP (Claim 4.3). By similar argument, we can show that each vertex is of degree at least 3 and $x(N[e]) > 2$ for every hyperedge e in H (Claim 4.4). Using these properties of the graph H, we show by case analysis that two tight Fano planes share at most 1 vertex (Lemma 4.5). From Lemma 4.5 we argue by case analysis that there are many edges intersecting a tight Fano plane which would contribute extra degrees (Lemma 4.6), so there are more non-zero variables than the number of tight constraints, contradicting that the solution is basic. Therefore we can show that tight Fano planes are disjoint (Lemma 4.7), by counting the total degree in a connected Fano plane component. Finally, we show that Fano planes do not exist at all in H. By applying the following result of Füredi [15], we get the conclusion that the integrality gap is at most 2.

Theorem 4.2 (Füredi)**.** *If H is a* 3*-uniform hypergraph which does not contain a Fano plane, then the integrality gap of* (LP) *for the hypergraph matching problem is at most two.*

Let $M(H)$ be a maximum matching in H. Since H is a counterexample to Theorem 1.4, there exists a basic fractional solution x to Fano-LP of H such that the integrality gap is greater than two, i.e. $x(E(H)) > 2|M(H)|$. First we argue that in x, every hyperedge e has fractional value $0 < x_e < \frac{1}{2}$.

Claim 4.3. *Let H be a minimal counterexample to* Theorem 1.4 *and x is a fractional solution to Fano-LP of H with integrality gap greater than two. Then $0 < x_e < \frac{1}{2}$ for every hyperedge e in H.*

[1]Actually we can write the stronger constraint $x(P) \leq 1$ for each Fano plane, but it is easier to analyze the weaker constraints using our method.

Proof. Suppose $x_e = 0$. Consider $H - e$, in which we remove the hyperedge e from H. Since H is a minimal counterexample, we have $x(E(H)) = x(E(H-e)) \leq 2|M(H-e)| \leq 2|M(H)|$, contradicting H is a counterexample. Therefore $x_e > 0$ for every hyperedge e in H. Suppose $x_e \geq \frac{1}{2}$. Let $e = \{v_1, v_2, v_3\}$. Consider $H' = H - v_1 - v_2 - v_3$ in which we remove v_1, v_2, v_3 and the hyperedges in $N[e]$ from H. Since $x_e \geq \frac{1}{2}$, we have $x(N[e]) \leq 2$. Therefore we have $x(E(H)) \leq x(E(H')) + 2 \leq 2|M(H')| + 2 \leq 2|M(H)|$, contradicting H is a counterexample. The first inequality follows from that $x(N[e]) \leq 2$, while the second inequality follows because H is a minimal counterexample, and the final inequality follows because $M(H') + e$ is a matching in H. $\square$

Similarly, we can argue that every vertex in H is of degree at least 3 and $x(N[e]) > 2$ for every hyperedge e.

Claim 4.4. *Every vertex in H is of degree at least 3, and $x(N[e]) > 2$ for every hyperedge e in H.*

Proof. We show that if there is a vertex of degree at most 2, then there is an edge with $x(N[e]) \leq 2$ and we can apply the argument in the proof of the previous claim to obtain a contradiction. Let v be a vertex of degree 2 and e, f be edges incident to it with $x_e \geq x_f$. Let $e = \{v, v_1, v_2\}$. Then $x(N[e]) \leq x_e + x_f + x(\delta(v_1) - e) + x(\delta(v_2) - e) \leq x_e + x_e + (1 - x_e) + (1 - x_e) = 2$. Omit f for the case when $\deg(v) = 1$. $\square$

A basic fractional solution is characterized by a set of tight inequalities. Let $\mathcal{D}$ be the set of tight degree constraints of the form $x(\delta(v)) = 1$ and let $\mathcal{P}$ be the set of tight Fano constraints of the form $x(P) = 2$. In a basic solution the number of nonzero variables is at most the number of tight constraints, i.e. $|E(H)| \leq |\mathcal{D}| + |\mathcal{P}|$. We will prove that if H has a Fano plane, then any basic solution will have a hyperedge e with $x_e \geq \frac{1}{2}$, contradicting that H is a minimal counterexample by Claim 4.3.

Suppose H has some Fano planes. In the following lemma we show that two tight Fano planes cannot share more than one vertex; otherwise there will be a hyperedge e with $x_e \geq \frac{1}{2}$. Note that this proof uses crucially the weaker constraints $x(P) \leq 2$, instead of the stronger constraints $x(P) \leq 1$. We are not able to argue that the tight Fano planes share at most one vertex if we use the stronger constraints, so the structure formed by the tight Fano planes could be more complicated. With the weaker constraints, there are only two ways the tight Fano planes can connect together (Lemma 4.7), which make the proof simpler.

Lemma 4.5. *Two tight Fano planes in H share at most one vertex.*

Proof. Let the two tight Fano planes be P and P'. We divide it into cases by the number of vertices shared:

1. 2 vertices: in this case the 2 Fano planes do not share any edges. The sum of degree constraints in $P \cup P'$ gives

$$12 \geq \sum_{v \in P \cup P'} \sum_{e \ni v} x_e \geq 3 \cdot \sum_{e \in P \cup P'} x_e = 12$$

so all vertices have to be tight and there cannot be any other edges incident to vertices in $P \cup P'$. By the Fano plane constraint, the sum of values of all 14 edges is 4. The two shared vertices will intersect 5 edges in each Fano plane and the sum of values of the 10 edges is strictly less than 2, by the degree constraint on the shared vertices. Therefore there must exist an edge of value at least $\frac{1}{2}$ in the remaining four edges, which is a contradiction.

2. 3+ vertices, not sharing any edges: the sum of degree constraints (≤ 11) would force at least one Fano plane constraint to be non-tight.

3. 3+ vertices, sharing at most 3 edges: if they share 3 edges, let them be a, b and c. The sum of x_a, x_b and degree constraints on the vertices of c gives

$$4 > x_a + x_b + 3 \geq x_a + x_b + \sum_{v \in c} \sum_{e \ni v} x_e = x_a + x_b + \sum_{e \in P \cup P'} x_e + 2 \cdot x_c > \sum_{e \in P} x_e + \sum_{e \in P'} x_e$$

so the Fano plane constraints cannot be both tight. The case for 1 or 2 edges shared is similar (leave out x_a or both x_a and x_b in the inequality above).

4. 3+ vertices, sharing 4 edges: in this case at least 6 vertices are shared. However, they cannot share 7 vertices because two Fano planes on the same set of vertices share at most 3 edges. Let a, b, c and d be the shared edges and u be the vertex in P that is not shared. The sum of degree constraints on the shared vertices gives

$$6 \geq \sum_{v \in P \setminus \{u\}} \sum_{e \ni v} x_e \geq 2 \cdot \sum_{e \in P \cup P'} x_e + (x_a + x_b + x_c + x_d).$$

Adding $x_a + x_b + x_c + x_d$ to both sides gives

$$8 > 2 \cdot \sum_{e \in P \cup P'} x_e + 2 \cdot (x_a + x_b + x_c + x_d) = 2 \cdot \sum_{e \in P} x_e + 2 \cdot \sum_{e \in P'} x_e$$

(since $x_a + x_b + x_c + x_d < 2$) so the Fano plane constraints cannot be both tight.

5. 3+ vertices, sharing 5+ edges: it is easy to see that they are essentially the same Fano plane. □

By Claim 4.4 every vertex is of degree at least 3. To show a contradiction we need to show that the total degree is greater than $3(|\mathcal{D}| + |\mathcal{P}|)$ since total degree is 3 times the number of non-zero edges and in a basic solution, we have $|E(H)| \leq |\mathcal{D}| + |\mathcal{P}|$. A vertex v in a Fano plane P is an *outgoing vertex* of P if v intersects edge(s) not belonging to P; note that an outgoing vertex is of degree at least 4. The following lemma shows that if there are many tight degree constraints in a Fano plane, the total degree of the vertices in this Fano plane must also be higher.

Lemma 4.6. *In a tight Fano plane in H, there are at least 4 outgoing vertices if all degree constraints are tight, and at least 3 outgoing vertices if 6 of the degree constraints are tight.*

Proof. In a Fano plane, the 7 edges contribute 21 to the total degree, so at least 7 of the constraints must be tight. If all 7 degree constraints are tight:

1. The tight Fano plane cannot be isolated (no outgoing vertices) because the sum of degree constraints would include every edge in the Fano plane thrice. This sum is 7 which means $\sum_{e \in P} x_e = \frac{7}{3}$ so the Fano plane constraint is violated.

2. If there is an edge e in the Fano plane that does not intersect any outgoing vertices and all vertices of e are tight, we can derive a contradiction. The sum of degree constraints on the vertices of e includes e thrice and all other edges once:

$$3 = \sum_{v \in e} \sum_{e' \ni v} x_{e'} = \sum_{e' \in P} x_{e'} + 2x_e$$

   Since $x_e < \frac{1}{2}$, the sum of values of the 7 edges would be strictly larger than 2 so the Fano plane constraint is violated. Since the smallest vertex cover for the Fano plane is three vertices of the same edge, this rules out the possibility of only 1 or 2 outgoing vertices, and if there are only 3 outgoing vertices, they must be on the same edge.

3. If there are 3 outgoing vertices on the same edge e, then we consider the tight constraint on the remaining 4 vertices: the sum of them (which is 4) includes every remaining edge in the Fano plane twice (Figure 4.1(a)). By the Fano plane constraint this means $x_e = 0$ contradicting $x_e > 0$.

If 6 of the degree constraints are tight:

4. The tight Fano plane cannot be isolated. The reason is similar to Case 1 above, with the sum replaced by $6+x$ where $0 < x < 1$ is the sum of values at the non-tight vertex.

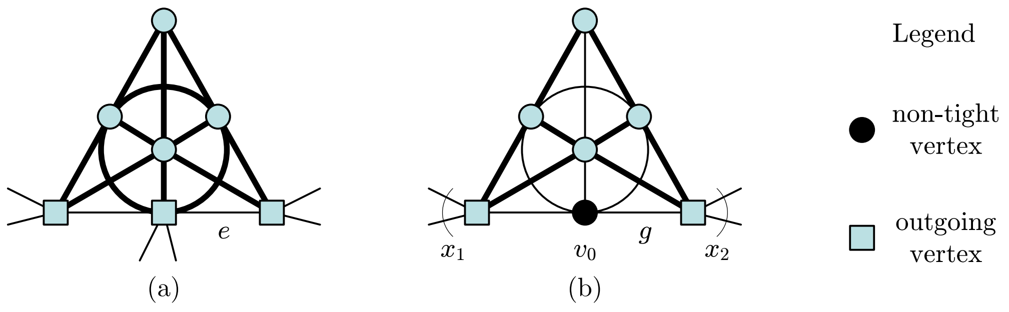


Figure 4.1: Lemma 4.6: (a) (Case 3) The sum of degree constraint on the round vertices includes every bold edge twice. (b) (Case 5, 6) To cover the edges that consist of only tight vertices (bold edges), at least 2 outgoing vertices are required.

5. When there is one non-tight vertex, there are 4 edges in the Fano plane that consist only of tight vertices (Figure 4.1(b)). To cover them with outgoing vertices as stated in Case 2 above, at least two vertices are required.

6. The case of two outgoing vertices is not possible either. Let the edges in the Fano plane be a, b, $\dots$, g and the non-tight vertex be v_0. Let e, f and g be the three edges incident to v_0. Case 2 above mandates that the two outgoing vertices must intersect the remaining four edges (a, b, c and d) so v_0 must be on the same edge (say, g) with the two outgoing vertices. Let the sum of values of edges outside the Fano plane that intersect the outgoing vertices be x_1 and x_2 respectively (Figure 4.1(b)).

   The sum of values incident to all vertices in the Fano plane give $(6 + x_e + x_f + x_g)$; this is equal to 3 times the Fano plane constraint plus x_1 and x_2. Therefore $x_1 + x_2 = x_e + x_f + x_g$.

   The sum of constraints on the two outgoing vertices give $x_a + x_b + x_c + x_d + 2x_g + x_1 + x_2 = 2$. Using the above substitution we have $x_a + x_b + x_c + x_d + x_e + x_f + 3x_g = 2$. Subtracting this by the Fano plane constraint, we get $x_g = 0$ which is a contradiction. □

Lemma 4.5 shows that two tight Fano planes share at most one vertex. In the following we prove a stronger claim that two tight Fano planes must also be disjoint. The proof is to consider a connected component of the tight Fano planes, and use Lemma 4.6 to argue that the total degree in this component is larger than thrice the number of tight constraints in this component. The proof uses an argument similar to Lemma 10 of Goemans [18].

Lemma 4.7. *Tight Fano planes in H are disjoint.*

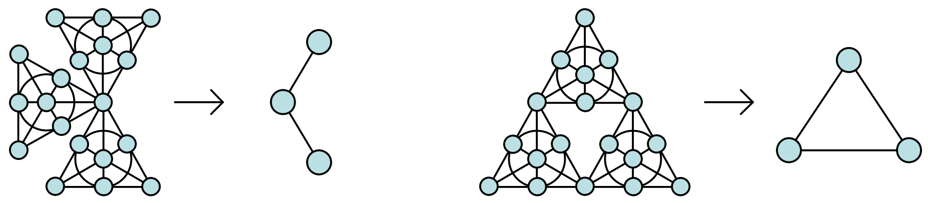


Figure 4.2: Construction of the graph G_v

Proof. Construct a graph G_v in which the vertex set is the set of tight Fano planes. If in the original graph a vertex is shared among a set K of k Fano planes, connect the vertices in K with a tree. By Lemma 4.5, two tight Fano planes can share at most one vertex, so the graph G_v is simple (Figure 4.2).

Consider a connected component C of G_v. Let there be m edges and n vertices in C. The number of vertices represented by C in the original graph is $7n - m$. Since they do not intersect other tight Fano planes, 3 times the number of tight degree constraints and Fano plane constraints has to be at least the total degree in the Fano planes. Therefore $3 \cdot (7n - m) + 3 \cdot n \geq 3 \cdot 7n$, and so $n \geq m$. Also, $m \geq n - 1$ because C is connected. Therefore C is either a tree or a unicyclic graph.

We need to show that total degree is strictly larger than $3(|\mathcal{P}| + |\mathcal{D}|)$. Total degree contributed by edges in the Fano planes is $3 \cdot 7n$, and if there are too many "*extra degree*" from edges outside the Fano planes, we get the desired contradiction. First we claim that a Fano plane receives at least 2 extra degrees if degree of the corresponding vertex in G_v is 1. Let the Fano plane be P. By Lemma 4.6, there are at least 3 outgoing vertices in P. Since it intersects with other Fano planes only at one vertex, there would be at least two other outgoing vertices. Therefore it receives at least 2 extra degrees.

If $m = n - 1$, the number of tight constraints is at most $7n + 1$ (n tight Fano plane constraints and at most $6n + 1$ tight degree constraints). Therefore to obtain a contradiction we need at least 4 extra degrees. Since there are at least two leaves in a tree, there are enough extra degrees to obtain the desired contradiction.

If $m = n$, the number of tight constraints is at most the number of edges. In this case any extra degree would lead to a contradiction. If in G_v there is a degree 1 vertex, then we get 2 extra degrees; otherwise, G_v is a cycle. By a similar argument as in the claim above, every Fano plane shares two vertices with other Fano planes. Therefore Lemma 4.6 suggests that each Fano plane receives at least one extra degree. □

Suppose that in x there exist some tight Fano planes. Lemma 4.7 shows that tight Fano planes are disjoint. By Lemma 4.6 we can see that the total degree is greater than thrice the total number of tight constraints. Hence there are no tight Fano planes. Then the number of tight (degree) constraints in H is at most $|V(H)|$, the number of vertices. By Claim 4.4 the number of hyperedges in H is at least $|V(H)|$. Therefore every vertex is of degree 3. If there is a Fano plane P in H, there would be no outgoing vertices, and thus $x(N[e]) < 2$ for any hyperedge e in P, but this contradicts Claim 4.4. Hence there are no Fano planes in H. In this case the result of Füredi shows that $x(E(H)) \leq 2|M(H)|$ for any fractional solution to the Fano linear program for the hypergraph matching problem.

Concluding Remarks

In this work the standard LP relaxation of the k-Dimensional Matching problem is studied and a polynomial time $(k-1)$-approximation algorithm is given, matching the integrality gap of the LP. We also studied the strengthening of the standard relaxation by adding the Fano plane constraint to give a better integrality gap for unweighted 3-Set Packing; however, no rounding algorithm is given.

A question to ask is whether we can write better LP with smaller integrality gap. In [12] we showed that Sherali-Adams relaxation, which is a hierarchy to obtain tighter relaxations in rounds, has gap of at least $k-2$ after $\Omega(\frac{n}{k^3})$ rounds (It is known that the polytope becomes integral after n rounds). On the other hand, by adding a type of constraint called *intersecting family constraint*, which mandates a fractional solution of at most one for each set of pairwise intersecting hyperedges, the resulting LP would have an integrality gap of no more than $\frac{k+1}{2}$. The integrality gap is smaller than that obtained by Sherali-Adams by a constant factor. It would be interesting to see an integrality gap example matching the analysis or improve the integrality gap analysis.

The algorithm for weighted k-Dimensional Matching used iterative rounding together with local ratio. This is a new way of analyzing the LP of a packing problem. It would be nice to see this technique applied to solve other packing problems.

Bibliography

[1] R. Aharoni: *Ryser's Conjecture for Tripartite 3-graphs.* Combinatorica **21**(1), 1–4, 2001.

[2] R. Aharoni, P. Haxell: *Hall's Theorem for Hypergraphs.* Journal of Graph Theory **35**, 83–88, 2000.

[3] E.M. Arkin, R. Hassin: *On Local Search for Weighted k-Set Packing.* Mathematics of Operations Research **23**(3), 640–648, 1998.

[4] V. Bafna, B. Narayan, R. Ravi: *Nonoverlapping Local Alignments (Weighted Independent Sets of Axis-parallel Rectangles).* Discrete Applied Math. **71**(1–3), 41–53, 1996.

[5] R. Bar-Yehuda, K. Bendel, A. Freund, D. Rawitz: *Local Ratio: A Unified Framework for Approximation Algorithms. In Memoriam: Shimon Even 1935–2004.* ACM Computing Surveys **36**(4), 422–463, 2004.

[6] R. Bar-Yehuda, M.M. Halldórsson, J.(S.) Naor, H. Shachnai, I. Shapira: *Scheduling Split Intervals.* SIAM Journal on Computing **36**(1), 1–15, 2006.

[7] P. Berman: *A $d/2$ Approximation for Maximum Weight Independent Set in d-Claw Free Graphs.* Proceedings of the 7th Scandinavian Workshop on Algorithms Theory (SWAT). Lecture Notes in Computer Science, Springer, Volume 1851, 31–40, 2000.

[8] P. Berman, P. Krysta: *Optimizing Misdirection.* Proceedings of the 14th Annual Symposium on Discrete Algorithms (SODA), 192–201, 2003.

[9] B. Bollobás: *Extremal Graph Theory.* Courier Dover Publications, 2004.

[10] M. Cai, X. Deng, W. Zang: *An Approximation Algorithm for Feedback Vertex Sets in Tournaments.* SIAM J. Computing **30**(6), 1993–2007, 2001.

[11] B. Chandra, M.M. Halldórsson: *Greedy Local Improvement and Weighted Set Packing Approximation.* Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 169–176, 1999.

[12] Y.H. Chan, L.C. Lau: *On linear and semidefinite programming relaxations for hypergraph matching.* Manuscript, 2009.

[13] J. Edmonds: *Paths, Trees, and Flowers.* Canadian Journal of Mathematics **17**, 449–467, 1965.

[14] J. Edmonds: *Maximum Matching and a Polyhedron with 0,1-vertices.* Journal of Research, National Bureau of Standards, Section B **69**, 125–130, 1965.

[15] Z. Füredi: *Maximum Degree and Fractional Matchings in Uniform Hypergraphs.* Combinatorica **1**(2), 155–162, 1981.

[16] Z. Füredi, J. Kahn, P.D. Seymour: *On the fractional Matching Polytope of a Hypergraph.* Combinatorica **13**(2), 167–180, 1993.

[17] M. Garey, D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., San Francisco, 1979.

[18] M.X. Goemans: *Minimum Bounded Degree Spanning Trees.* Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 273–282, 2006.

[19] J. Håstad: *Clique is Hard to Approximate within $n^{1-\epsilon}$.* Acta Math **182**(1), 105–142, 1999.

[20] M.M. Halldórsson: *Approximations of Weighted Independent Set and Hereditary Subset Problems.* Proceedings of the 5th Annual International Conference on Computing and Combinatorics. Lecture Notes in Computer Science, Springer, Volume 1627, 261–270, 1999.

[21] M.M. Halldórsson, J. Kratochvíl, J.A. Telle: *Independent Sets with Domination Constraints.* Proceedings of the 25th International Colloquium on Automata, Languages and Programming. Lecture Notes in Computer Science, Springer, Volume 1443, 176–185, 1998.

[22] E. Hazan, M. Safra, O. Schwartz: *On the Complexity of Approximating k-Set Packing.* Computational Complexity **15**(1), 20–39, 2006.

[23] C.A.J. Hurkens, A. Schrijver: *On the Size of Systems of Sets Every t of which have an SDR, with an Application to the Worst-case Ratio of Heuristics for Packing Problems.* SIAM Journal on Discrete Mathematics **2**, 68–72, 1989.

[24] K. Jain: *A Factor 2 Approximation Algorithm for the Generalized Steiner Network Problem.* Combinatorica **21**(1), 39–60, 2001.

[25] D. König: *Graphok és Matrixok* [Hungarian; *Graphs and Matrices*]. Matematikai és Fizikai Lapok **38** 116–119, 1931.

[26] C.W.H. Lam: *Search for a Finite Projective Plane of Order 10.* American Mathematical Monthly **98**(4) 305–318, 1991.

[27] G.J. Minty: *On Maximal Independent Sets of Vertices in Claw-free Graphs.* Journal of Combinatorial Theory Series B **28**(3), 284–304, 1980.

[28] D. Nakamura, A. Tamura: *A Revision of Minty's Algorithm for Finding a Maximum Weight Stable Set of a Claw-free Graph.* Journal of the Operations Research Society of Japan **44**(2), 194–204, 2001.

[29] M. Singh, L.C. Lau: *Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal.* Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), 661–670, 2007.

[30] A. Schrijver: *Combinatorial Optimization: Polyhedra and Efficiency.* Springer-Verlag, Berlin, 2003.