Approximating the Maximum Acyclic Subgraph

by

Alantha Newman

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Author .....
Department of Electrical Engineering and Computer Science
Spring, 2000

Certified by .....
Santosh Vempala
Assistant Professor of Mathematics
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Approximating the Maximum Acyclic Subgraph

by

Alantha Newman

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2000, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

In this thesis, we study the maximum acyclic subgraph problem: Given a directed graph $G = (V, E)$, find a maximum cardinality subset $E' \subseteq E$ such that $G = (V, E')$ is acyclic. The maximum acyclic subgraph problem is an NP-hard optimization problem for which the best-known approximation guarantee is 2. We show that the maximum acyclic subgraph problem cannot be approximated to within $\frac{65}{66} + \epsilon$ for any $\epsilon > 0$. The reduction is from Håstad's maximum satisfiability of linear equations modulo 2 with three variables per equation. It is already known that the integrality gap of two basic linear programming relaxations is 2, but we formalize this here since it is not recorded elsewhere. We also study graphs in which the maximum degree is 3 and show that if for any $\epsilon > 0$ there exists a $(17/18+\epsilon)$-approximation algorithm for the maximum acyclic subgraph problem in such graphs, then there is a $\delta > 0$ such that there is a $(1/2 + \delta)$-approximation algorithm for the maximum acyclic subgraph problem in general graphs. We give a $\frac{8}{9}$-approximation algorithm for the maximum acyclic subgraph problem in graphs with maximum degree 3.

Thesis Supervisor: Santosh Vempala
Title: Assistant Professor of Mathematics

Acknowledgments

I thank Santosh Vempala for being my advisor and for patiently teaching me so many new things. I also thank John Dunagan for enthusiastically answering many questions related to this thesis throughout the year and for showing me the construction is Section 3.3.

Contents

Chapter 1

Introduction

Many important and practically motivated optimization problems are NP-hard. Unless P = NP, we cannot efficiently find exact solutions to these problems. One option is to relax the requirement that the size (or cost) of the solution is optimal. Instead, we can look for a solution whose size if provably within some factor of the size of an optimal solution. In this scenario, we want to find an algorithm that yields the best possible factor or approximation ratio. See [11] for an overview of approximation algorithms.

1.1 The Problem

In this paper, we explore the approximability and inapproximability of a particular NP-hard optimization problem known as the *maximum acyclic subgraph problem.* The maximum acyclic subgraph problem is defined as follows: Given a directed graph $G = (V, E)$, find a maximum cardinality subset $E' \subseteq E$ such that $G = (V, E')$ is acyclic. This problem is an example of an NP-hard optimization problem for which a simple approximation algorithm offers the best-known guarantee on the size of the solution. For any linear ordering of the vertices, the set of forward edges or the set of backward edges contains at least half the edges. Each set is acyclic. The entire set of edges, E, is an upper bound on the size of an optimal solution. Therefore, the larger of these sets yields a $\frac{1}{2}$-approximation.

An outstanding open problem is: Can we do better than half of the optimum?

1.2 Overview of this Thesis

In Chapter 2, we show that it is NP-hard to approximate the maximum acyclic subgraph to within $\frac{65}{66} + \epsilon$ for any $\epsilon > 0$. This means that if we could find an algorithm with an approximation guarantee of $\frac{65}{66} + \epsilon$ for some $\epsilon > 0$, then we could solve the problem exactly. We give a reduction from 3-SAT to the maximum acyclic subgraph problem to illustrate

the idea behind the inapproximability reduction. We then give a reduction from Håstad's linear equations modulo 2 with 3 variables per equation, which, although more complicated, yields a better inapproximability constant.

A common tool for finding approximation algorithms for NP-hard optimization problems is linear programming. In Chapter 3, we discuss linear programming relaxations for the maximum acyclic subgraph problem. The basic linear programming relaxation for the maximum acyclic subgraph problem has an integrality gap of 2, i.e. there is a class of graphs for which the true maximum is very close to half the edges, while the linear programming relaxation returns a fractional solution with value very close to the size of the entire edge set. Therefore, we cannot use the basic linear programming relaxation to obtain a better-than-half approximation.

In Chapter 4, we first investigate the maximum acyclic subgraph problem restricted to Eulerian graphs. We present a theorem due to Lovász and Chen [8] showing that the general problem can be reduced to this restricted problem. We next investigate the maximum acyclic subgraph problem restricted to graphs with maximum degree 3, i.e. in-degree plus out-degree of any vertex in the graph is at most 3. The maximum acyclic subgraph problem remains NP-hard for such graphs [7]. We show that there is a connection between the approximability of degree-3 graphs and general graphs. In particular, we show that if for any $\epsilon > 0$ there exists a $(\frac{17}{18}+\epsilon)$-approximation algorithm for the maximum acyclic subgraph problem in degree-3 graphs, then there exists a better-than-half approximation algorithm for the maximum acyclic subgraph problem in general graphs.

Finally, in Chapter 5 we show that we can actually approximate the maximum acyclic subgraph in degree-3 graphs to within $\frac{8}{9}$.

Chapter 2

Inapproximability

In this chapter, we will describe two reductions which essentially use the same idea. The first reduction is from 3-SAT and the second reduction is from Håstad's linear equations modulo 2 with 3 variables per equation. The idea behind both reductions is that an assignment that satisfies a particular clause corresponds to removing one less edge from the representative clause gadget than an than an assignment that does not satisfy this clause.

2.1 Reduction from 3-SAT

In this section we will give an approximation-preserving reduction from MAX-3-SAT to the maximum acyclic subgraph problem. MAX-3-SAT is defined as the following problem: Given a CNF formula with at most 3 variables per clause, find an assignment of the variables that maximizes the number of satisfied clauses.

2.1.1 The Construction

Given a 3-SAT formula F with n variables and m clauses, we construct a corresponding multigraph G using the following rules: (We assume every variable in F appears at least once negated and once unnegated.)

1. For each variable $x \in F$, we create 2 vertices x_1 and x_2. These two vertices will form the *variable* gadget for the variable x.

2. For each clause $C_k \in F$, we create a directed 6-cycle and label each of 3 alternating edges with a distinct literal from the clause C_k. This will be the *clause* gadget for the clause C_k.

3. Each clause gadget is linked up to the variable gadgets as follows.

- For an edge (i, j) labeled x in a clause gadget, we add a directed edge from vertex j to vertex x_1, an edge from x_1 to x_2, and an edge from x_2 to i.
- For an edge (i, j) labeled $\overline{x}$, we add a directed edge from vertex j to vertex x_2, an edge from x_2 to x_1, and an edge from x_1 to i.

Note that the graph G has $15m$ edges in all — $6m$ for the equation gadgets, $6m$ for connecting the equation gadgets to the variable gadgets and $3m$ for the variable gadgets (one edge per occurrence).

2.1.2 The Proof

The following theorem will be our starting point.

Theorem 1 (Håstad [6]) *For every $\epsilon > 0$, it is NP-hard to tell if a given 3-SAT formula is satisfiable or at most $m(\frac{7}{8} + \epsilon)$ of its clauses are satisfiable.*

We prove:

Theorem 2 *The maximum acyclic subgraph cannot be approximated to within $\frac{95}{96} + \epsilon$ for any $\epsilon > 0$.*

The proof of Theorem 2 will use the lemmas below.

Lemma 1 *A minimal feedback arc set is acyclic.*

Proof. An acyclic graph can be viewed as an ordering of the vertices such that all the arcs are in the forward direction, i.e. for each arc (i, j), i comes before j in the ordering. Given a feedback arc set, consider such an ordering for the acyclic graph obtained upon deleting the feedback arc set. If the feedback arc set has any edges in the forward direction, then it is not minimal (such an edge can be added to the acyclic graph without creating any cycles). Thus the feedback arc set consists only of backward edges and hence is itself acyclic. □

Lemma 2 *The minimum feedback arc set of G either has all the edges from x_{i1} to x_{i2} and none of the edges from x_{i2} to x_{i1} or vice versa, for all i.*

Proof. If we include any edges from x_{i1} to x_{i2} and even one edge from x_{i2} to x_{i1} in the minimum feedback arc set, then it would not be acyclic, which is a contradiction to Lemma 1. If we don't include all the edges from one of the sets in the minimum feedback arc set, then we will not have an edge from every cycle in the minimum feedback arc set, which is also a contradiction. □

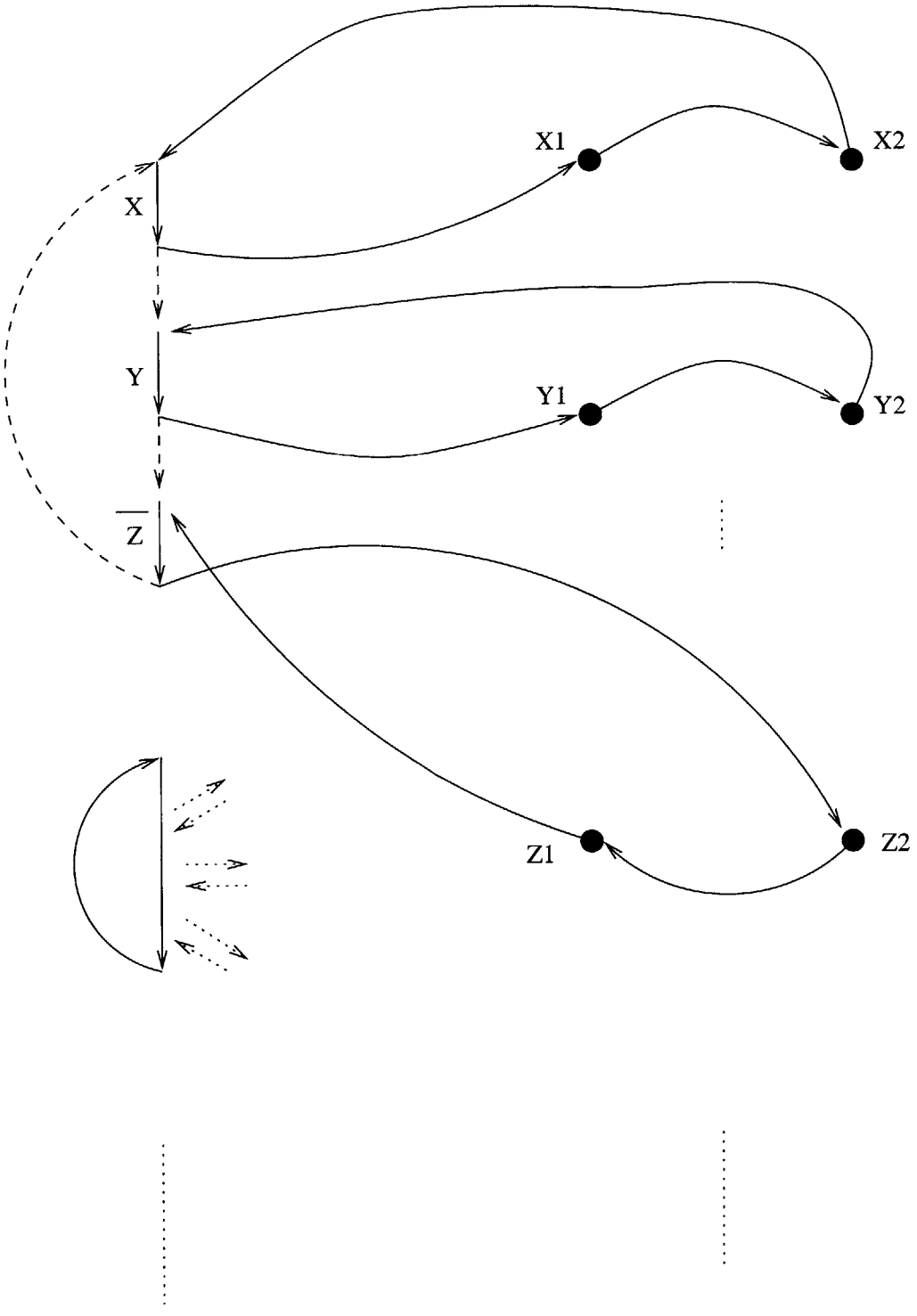


Figure 2-1: The clause and variable gadgets.

Lemma 3 *The minimum feedback arc set for the graph G contains $3m + u$ edges, where u is the minimum number of unsatisfied clauses of the formula F.*

Proof. The theorem will be proved as a consequence of the following two claims: (i) Given an assignment for the variables in F that results in u unsatisfied clauses, we can construct a Feedback Arc Set of size at most $3m+u$. (ii) Conversely, given a Feedback Arc Set of size $3m + u$, we can find an assignment for the variables of F such that no more than u clauses are unsatisfied.

First, we show that if we have a set of arcs including all the arcs from x_{i1} to x_{i2} or vice-versa for all i, and at least one edge from each of the three 4-cycles (linking the literals arcs to their variable gadgets) and the 6-cycle in every gadget, then the resulting set of edges is a Feedback Arc Set. We will do this by showing that the leftover edges form an acyclic graph.

If for every clause gadget, each 4-cycle and 6-cycle is missing at least one edge, then the set of remaining edges from each clause gadget is acyclic. We still need to show that there are no cycles that include edges from multiple gadgets. Suppose we take a walk on the graph G. We start the walk on some edge labeled x_i and go to vertex x_{i1}. If edge x_i is not in the Feedback Arc Set, then edge (x_{i1}, x_{i2}) must be, which means edge (x_{i2}, x_{i1}) is not, so all edges labeled $\overline{x}_i$ are included in the Feedback Arc Set. Therefore, when we try to depart from vertex x_{i1} and move to a different gadget, we will not be able to do so, because all edges from this vertex lead to edges labeled $\overline{x}_i$, which have been included in the Feedback Arc Set. So the resulting graph is acyclic.

(i) Given an assignment for the variables in F, we will show that we can find a feedback arc set including exactly 3 arcs for each satisfied clause and exactly 4 arcs from each unsatisfied clause. We construct the feedback set as follows. If x_i is set to TRUE, then we include all the arcs from x_{i2} to x_{i1} in the feedback set; if it is set to FALSE, we include all the arcs from x_{i1} to x_{i2}. Then we include all the arcs in the clause gadgets that correspond to true literals. In addition, we include one arc corresponding to a literal from each clause gadget for which all the literals are false. The resulting subset of arcs is a feedback set by the argument in the previous paragraph and has a total of $3m + u$ arcs.

(ii) Given a feedback arc set, we now show how to construct an assignment from it. First we delete edges from the feedback arc set until it is minimal. Then we assign each variable x_i in F a value depending on which set of edges with endpoints in $\{x_{i1}, x_{i2}\}$ is included in the feedback arc set. If all the edges from x_{i2} to x_{i1} are in the feedback arc set, then the variable x_i is set to TRUE. Otherwise all the edges from x_{i1} to x_{i2} are in feedback arc set, and then x_i is set to FALSE.

Every clause gadget that has only 3 associated arcs (i.e. arcs in the 6-cycle or in the 4-cycles linked to the literals in the clause) in the feedback arc set must have at least 1 arc from the 6-cycle in the feedback arc set, so at least one of the literals in the clause will have been assigned to TRUE. Thus any clause that is false has 4 arcs included in the feedback arc set. So if the feedback arc set has $3m + u$ arcs, the assignment leaves at most u clauses unsatisfied. □

Corollary 4 follows from Lemma 3 and the fact that G has $15m$ edges.

Corollary 4 *The Maximum Acyclic Subgraph for G is of size $12s + 11u$ where s and u represent the number of satisfied and unsatisfied clauses, respectively, for an assignment that satisfies the maximum number of clauses.*

Proof of Theorem 2. Using Corollary 4 and the fact that, given a 3-SAT formula with m clauses, it is NP-hard to distinguish between an assignment that satisfies $(\frac{7}{8} + \epsilon)m$ of the clauses and an assignment that satisfies m clauses (Theorem 1), we see that it is NP-hard to distinguish between a graph that has a maximum acyclic subgraph of size $12(\frac{7}{8} + \epsilon)m + 11(\frac{1}{8} - \epsilon)m$ and a graph that has a maximum acyclic subgraph of size $12m$. If we could approximate the maximum acyclic subgraph to within $\frac{95}{96} + \epsilon$, then we could distinguish between these two cases. Therefore, it is NP-hard to approximate the maximum acyclic subgraph to within $\frac{95}{96} + \epsilon$. □

2.2 Reduction from Linear Equations Modulo 2

In this section, we will give an approximation-preserving reduction from linear equations modulo 2 with three variables to the maximum acyclic subgraph problem.

2.2.1 The Construction

Given a set of m linear equations on n variables, we construct a graph G using the following rules: (We assume all equations have the right hand size zero by negating one literal if necessary.)

1. For each variable $x \in F$, we create two vertices and two edges. The vertices are x_0 and x_1 and the edges are (x_1, x_0) and (x_0, x_1). These vertices and edges will form the *variable* gadget.

2. For each clause $C_j \in F$, we add the *clause* gadget. The clause gadget is shown in Figure 2-2. For a literal x in the clause we create a 4-cycle $\{x_2, x_3, x_4, x_5\}$. We label edge (x_5, x_2) as $x = 1$ and edge (x_3, x_4) as $x = 0$. We do this for each literal in the clause.

Then we add the following 12 edges: $(z_2, x_5), (z_2, y_3), (z_4, y_3), (z_4, x_3), (x_2, z_3), (x_2, y_5), (x_4, z_5), (x_4, y_5), (y_2, z_3), (y_2, z_5), (y_4, x_3), (y_4, x_5)$

3. We connect each clause gadget to the corresponding variable gadgets in the following way: For a literal x, we connect the corresponding 4-cycle in the clause gadget to the variable gadget by adding edges $(x_2, x_1), (x_1, x_3), (x_0, x_5), (x_4, x_0)$. The idea is that an edge in the clause gadget that corresponds to a variable x being set to 1 (labeled $x = 1$ in Figure 2-2) should be in a cycle with the edge in the variable gadget that corresponds to the variable being set to 0 (labeled 0 in Figure 2-2), so that one of these edges is removed and the settings of the variable is determined by the remaining edge. Since all clauses are connected to the same variable gadget, this will maintain consistency in the variable assignments of each clause.

The resulting graph G has $36m + 2n$ edges, 36 for each clause gadget and 2 for each variable gadget. In order to relate variable assignments to acyclic subgraphs of G, we will say removing the edge (x_1, x_0) corresponds to setting the variable x to true, and removing the edge (x_0, x_1) corresponds to setting the variable to false.

2.2.2 The Proof

We will use another theorem of Håstad for this proof:

Theorem 3 (Håstad [6]) *For every $\epsilon > 0$, it is NP-hard to tell if a given a set of linear equations modulo 2 with 3 variables is satisfiable or at most $m(\frac{1}{2} + \epsilon)$ of its clauses are satisfiable.*

Lemma 5 *The minimum feedback arc set for the graph G contains $n + 3m + u$ edges, where u is the minimum number of unsatisfied equations.*

Proof. By Lemma 1, exactly one edge from every variable gadget is in the minimum feedback arc set. In addition, we need to show the following things:

(i) Given a clause, $x + y + z = 0$, and an assignment that satisfies this clause, we will need to remove only three edges from the corresponding clause gadget so that the subgraph consisting of the clause gadget and it three corresponding variable gadgets is acyclic. There are four assignments to the variables x, y, z that satisfy this clause. They are: $\{\{0,0,0\}, \{0,1,1\}, \{1,1,0\}, \{1,0,1\}\}$. We need to show that for any one of these assignments, if we remove the three corresponding edges, then the subgraph of the remaining edges from the clause gadget and relevant variable gadgets is acyclic.

First, note that any cycle in a clause gadget must contain labeled edges. This is because for every non-labeled edge (i, j), i is a vertex such that the only incoming edge is a labeled edge, and j is a vertex such that the only outgoing edge is a labeled edge.

We now consider the four possible satisfying assignments for the clause $x + y + z = 0$:

1. $\{x = 0, y = 0, z = 0\}$. This means that all edges in Figure 2-2 labeled $x = 1, y = 1, z = 1$ are removed. The three edges labeled $x = 0, y = 0, z = 0$ are not in a cycle together. If they were, vertex x_4 would have to be in a cycle, but all four of its out edges lead to edges labeled 1, which are removed. Thus, the remaining set of edges is acyclic.

2. $\{x = 0, y = 1, z = 1\}$. Then the edges labeled $x = 1, y = 0, z = 0$ are removed. The remaining graph is acyclic, since if you consider the four edges going out from the edge labeled $z = 1$, all four of these edges lead to a vertex whose only out edge has been removed.

3. $\{x = 1, y = 0, z = 1\}$. Then the edges labeled $x = 0, y = 1, z = 0$ are removed. If we consider vertex x_2, which is the endpoint of the edge labeled $x = 1$, all four edges lead to vertices whose only out edge has been removed. So the remaining set of edges is acyclic.

4. $\{x = 1, y = 1, z = 0\}$. We remove edges $x = 0, y = 0, z = 1$. Consider vertex z_4, which is the endpoint of the edge labeled $z = 0$. All four edges leaving this vertex lead to vertices, whose only out edge has been removed. So the remaining set of edges is acyclic.

(ii) Given a clause, $x + y + z = 0$, and an assignment that does not satisfy this clause, we will need to remove exactly four edges from the corresponding clause gadget so that the subgraph consisting of the clause gadget and its three corresponding variable gadgets is acyclic. There are four assignments to the variables x, y, z that do not satisfy this clause. They are: $\{\{1,1,1\}, \{0,0,1\}, \{1,0,0\}, \{0,1,0\}\}$.

1. $\{x = 1, y = 1, z = 1\}$. Then the edges labeled $x = 1, y = 1, z = 1$ remain and form a cycle. So we must remove one of these edges and the remaining graph is acyclic.

2. $\{x = 0, y = 0, z = 1\}$. Then the edges $x = 0, y = 0, z = 1$. These edges are in a cycle, so, again, we must remove of four edges.

3. $\{x = 1, y = 0, z = 0\}$. Then the edges $x = 1, y = 0, z = 0$ remain and create a cycle, so we must remove four edges.

4. $\{x = 0, y = 1, z = 0\}$. Then the edges $x = 0, y = 1, z = 0$ remain and form a cycle, so we must remove four edges.

(iii) For each variable gadget, if we remove one of the two edges and the corresponding edge from the clause gadgets representing clauses that contain this variable, then the remaining graph does not contain any cycle composed of edges from more than one clause gadget. For the clause $x + y + z = 0$, consider the edge (x_2, x_1). If a cycle contains this edge, it must also contain the only incoming edge to vertex x_2, which is the edge labeled $x = 1$. If these edges are contained in a cycle with edges from another clause gadget, then at vertex x_1, we can move to another gadget. However, we will arrive at a vertex such that the only out edge corresponds to the edge that remains iff x has been set to 0, which is not the case if the edge labeled $x = 1$ was present. So there cannot be any cycles that use edges from more than one clause gadget.

It follows from (i),(ii), and (iii) that the minimum feedback arc set has size n+3m+u.□

Corollary 6 *The Maximum Acyclic Subgraph for G is of size $n + 33s + 32u$ where s and u represent the number of satisfied and unsatisfied clauses, respectively, for an assignment that satisfies the maximum number of clauses.*

Theorem 4 *The maximum acyclic subgraph can not be approximated to within $\frac{65}{66}$ for any $\epsilon > 0$.*

Proof. By Corollary 6 and by Theorem 3 it is NP-hard to distinguish between a graph that has a maximum acyclic subgraph of size $n + 33(\frac{1}{2} + \epsilon)m + 32(\frac{1}{2} - \epsilon)m$ and a graph that has a maximum acyclic subgraph of size $n + 33m$. If we could approximate the maximum acyclic subgraph to within $\frac{2n+65}{2n+66} + \epsilon$, then we could distinguish between these two cases. Therefore it is NP-hard to approximate to approximate the maximum acyclic subgraph to within $\frac{2n+65}{2n+66} + \epsilon$. We can make n arbitrarily small compared to m by creating another set of linear equations in which each original equation appears k times for some k so that we have km clauses and only n variables. The ratio $\frac{2n+65}{2n+66}$ is arbitrarily close to $\frac{65}{66}$ as k becomes large. Therefore, it is NP-hard to approximate the maximum acyclic subgraph to within $\frac{65}{66} + \epsilon$. □.

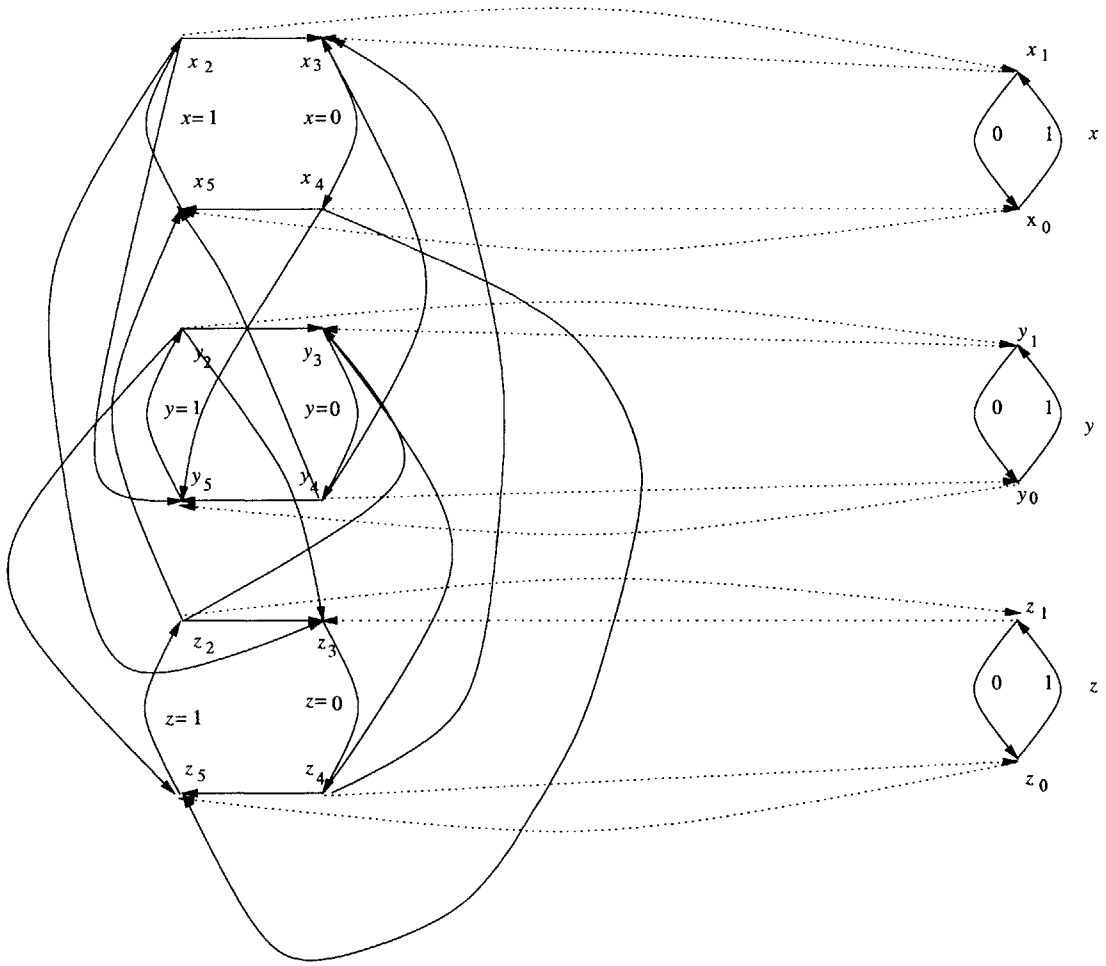


Figure 2-2: The clause and variable gadgets for $x + y + z = 0$.

Chapter 3

Linear Programming Relaxations

In this chapter we discuss linear programming relaxations for the maximum acyclic subgraph problem.

3.1 Two Integer Programs

The maximum acyclic subgraph problem can be viewed as maximizing the number of edges subject to a constraint for every cycle. The constraint specifies that the sum of the edge variables on a cycle of length C is at most $|C| - 1$. These constraints are described by the following integer program:

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{ij \in E} x_{ij} \\ &\text{subject to:} \\ &&& \textstyle\sum_{ij \in C} x_{ij} \leq |C| - 1 \\ &&& \forall_{ij \in E} \quad x_{ij} \in \{0, 1\} \end{aligned}$$

It is NP-hard to solve this integer program. However, we can relax the requirement that x_{ij} are in $\{0, 1\}$ and replace it with the requirement that that $0 \leq x_{ij} \leq 1$. We can solve the resulting linear program in polynomial time using the Ellipsoid Algorithm because it has the following polynomial separation oracle [5]. Given a solution to the linear program, we can consider the graph with each edge (i, j) assigned weight $1 - x_{ij}$. Then we can find the minimum weight cycle. If there is any cycle that has weight less than 1, then we have found a cycle with value more than $|C| - 1$ which is a violated constraint. We refer to this linear program as LP_1.

Another integer program for the maximum acyclic subgraph problem has a variable for every pair of vertices $i, j \in V$. In this program, there are only constraints for 2- and 3-cycles:

$$\text{maximize} \quad \textstyle\sum_{ij \in E} x_{ij}$$

subject to:

$$\begin{aligned} \forall_{i,j} \quad & x_{ij} + x_{ji} = 1 \\ \forall_{i,j,k} \quad & x_{ij} + x_{jk} + x_{ki} \leq 2 \\ \forall_{i,j} \quad & x_{ij} \in \{0, 1\} \end{aligned}$$

Lemma 7 *An integral solution to the above integer program represents an acyclic subgraph.*

Proof. Consider some acyclic subgraph $S \subseteq E$ that includes every edge which is assigned a 1 by this integer program. If each 2-cycle in the complete directed graph has at most one edge S and each 3-cycle has at most two edges in S, then we prove by induction that every cycle of length $|C| \geq 4$ has at most $|C| - 1$ edges in S.

Consider a 4-cycle $\{v_1, v_2, v_3, v_4\}$. Choose two non-adjacent vertices v_1 and v_3 on the 4-cycle. Consider the 2-cycle that connects these two vertices. One of these edges is in S, say (v_3, v_1). Then we cannot include both (v_1, v_2) and (v_2, v_3) in S, so at most three edges from this 4-cycle are in S. Similarly, assume each cycle of length at most C contains at least one edge assigned 0 by the integer program. Then consider some cycle $x = \{v_1, v_2, \cdots, v_{C+1}\}$ of length $C + 1$ and choose any two non-adjacent vertices on the cycle, v_i and v_j. Consider the 2-cycle that joins these vertices. One of the edges, say (v_i, v_j) is in S. Since the cycle $\{v_i, v_j, v_{j+1}, \cdots v_{i-1}\}$ has length at most C it follows that at least one of the edges in x on the path from x_j to x_i is assigned 0 by the integer program. Therefore, we can include at most C edges from a cycle of length $|C| + 1$ in S. □

Again, it is NP-hard to solve this integer program. However, we can solve the relaxation in polynomial time using the Ellipsoid Algorithm because there are only a polynomial number of constraints [5]. We will refer to this relaxation as LP_2.

3.2 Linear Program Integrality Gap

The integrality gap of a linear program is defined as the ratio of the size of the optimal integer solution to the size of the optimal fractional solution returned by the linear program. It is not known how to use a linear program to obtain an approximation ratio that is better than the integrality gap. In this section, we will show that the integrality gap for both LP_1 and LP_2 is 2 implying that these linear programs will not lead to a better-than-half approximation algorithm. As a basis for the construction, we will use the fact that there exists a class of undirected graphs with girth g and $\Theta(n \cdot (\frac{n}{8})^{\frac{1}{g}})$ edges. (See Section 3.3 for the construction.)

Given such a graph G with girth $\sqrt{\log n}$ (we could use any function that is $o(\log n)$ for the girth), we will show that the following lemma holds:

Lemma 8 *There exists at least one directed orientation of G such that for any ordering of the vertices, the number of back edges is at least $(1-\epsilon)\frac{|E|}{2}$ for any $\epsilon > 0$.*

Proof. For any $\epsilon > 0$, we show that with non-zero probability, a directed orientation of G has at least $(1-\epsilon)\frac{m}{2}$ back edges for any ordering of the vertices where m is the number of edges in G, i.e. the size of its maximum acyclic subgraph is very close to half the edges. For this proof, we use a Chernoff bound found in [9]:

Theorem 5 *Let $X_1, X_2, \cdots, X_n$ be independent Poisson trials such that, for $1 \leq i \leq n$, $\Pr[X_i = 1] = p_i$, where $0 < p_i < 1$. Then, for $X = \sum_{i=1}^{n} X_i$, $\mu = \mathrm{E}[X] = \sum_{i=1}^{n} p_i$, and $0 < \delta \leq 1$,*

$$\Pr[X < (1-\delta)\mu] < e^{-\frac{\mu\delta^2}{2}}$$

We fix an ordering of the vertices and we choose a directed orientation of G at random. For each edge (i,j), we direct the edge from i to j with probability $\frac{1}{2}$ and from j to i with probability $\frac{1}{2}$. Thus, each edge is a forward edge or a backward edge with equal probability. We associate an indicator random variable X_i with the i^{th} edge so that X_i is 1 if the i^{th} edge is a backward edge and 0 if it is a forward edge in this ordering. Thus, for all X_i, $p_i = \frac{1}{2}$. X is the random variable that represents the number of back edges and $\mu = \frac{m}{2}$, i.e. the expected number of back edges for a fixed ordering of the vertices is $\frac{m}{2}$. Using Theorem 5, we find that:

$$\Pr[X < (1-\epsilon)\tfrac{m}{2}] < e^{\frac{m\epsilon^2}{4}}$$

In words, the probability that a particular directed orientation of the edges with respect to a fixed ordering of the vertices has less than $(1-\epsilon)\frac{m}{2}$ back edges is exponentially small in m for any $\epsilon > 0$.

Now we fix the orientation of the edges and show that with non-zero probability this directed graph has close to $\frac{m}{2}$ back edges for every ordering of the vertices. There are $n! \leq 2^{n \log n}$ orderings of the vertices. By union bound, the probability that this directed graph has less than $(1-\epsilon)\frac{m}{2}$ back edges for *at least one* ordering of the vertices is:

$$\leq e^{-\frac{m\epsilon^2}{4}} 2^{n \log n}$$

If this quantity is < 1, then the probability that a particular orientation of the edges has at least $(1-\epsilon)\frac{m}{2}$ back edges for *every* ordering of the vertices will be > 0.

If $e^{-\frac{m\epsilon^2}{4}} 2^{n \log n} < 1$, then taking the log of each side, we have:

$$-\frac{m\epsilon^2}{4} + n \log n < 0 \quad \Rightarrow \quad n \log n < \frac{m\epsilon^2}{4} \quad \Rightarrow \quad \epsilon > \sqrt{\frac{4n \log n}{m}}$$

Let $m = \Theta(n \cdot (\frac{n}{8})^{\frac{1}{9}})$:

$$\epsilon > \sqrt{\frac{4 \log n}{(\frac{n}{8})^{\frac{1}{g}}}}$$

When $g = \sqrt{\log n}$, we need to show that ϵ is $o(1)$. Then for any fixed $\epsilon > 0$, this inequality will be true for large enough n. Since $(\frac{n}{8})^{\frac{1}{\sqrt{\log n}}} = 2^{\sqrt{\log n} - \frac{3}{\sqrt{\log n}}}$ and $\log n = 2^{\log \log n}$ and $\log \log n < \sqrt{\log n} - \frac{3}{\sqrt{\log n}}$ (since $\log x < \sqrt{x} - \frac{3}{\sqrt{x}}$), we have $\epsilon = o(1)$. Thus, with non-zero probability, there is some directed graph with girth $\sqrt{\log n}$ that has a maximum acyclic subgraph of size $(1 - \epsilon)\frac{|E|}{2}$ for any $\epsilon > 0$. □

We will refer to a directed graph on n vertices with girth $\sqrt{\log n}$ and maximum acyclic subgraph of size at least $(1 - \epsilon)\frac{m}{2}$ as G_n^*. We will use G_n^* to prove the following lemmas:

Lemma 9 *The integrality gap of LP_1 is 2.*

Proof. For any $\epsilon > 0$, we can find an n such that G_n^* has an maximum acyclic subgraph of size at least $(1 - \epsilon)\frac{m}{2}$. Since G_n^* has girth at least $\sqrt{\log n}$, a feasible solution for LP_1 is to assign each edge in E the value $(1 - \frac{1}{\sqrt{\log n}})$. Thus the solution of LP_1 has size at least $|E|(1 - \frac{1}{\sqrt{\log n}})$. The ratio of the optimal solution to the optimal fractional solution is $\frac{1 - \frac{1}{\sqrt{\log n}}}{\frac{1}{2}(1-\epsilon)}$. As ϵ decreases, n increases. Thus, $\lim_{n \to \infty} \frac{1 - \frac{1}{\sqrt{\log n}}}{\frac{1}{2}(1-\epsilon)} = 2$. Since this ratio can be made arbitrarily close to 2 for large n, the integrality gap is 2. □

Lemma 10 *The integrality gap of LP_2 is 2.*

Proof. Given a maximal solution to LP_1 for a graph G, we can construct a solution to LP_2 for G with the same objective value. A solution to LP_1 includes an assignment for every edge in G. The solution we will construct will contain an assignment for every pair of vertices $i, j \in V$. Let x be the maximal solution given for LP_1.

We extend the solution x as follows: for all $(i, j) \in E, (j, i) \notin E$, we assign $x_{ji} = x_{ij}$. Now every cycle in x of length $|C|$ has total value at most $|C| - 1$. This is equivalent to saying that every cycle has total value at least 1, since if some cycle has total value less than 1, then the cycle composed of the complementary edges must have total value more than $|C| - 1$. This is clearly true of all cycles in E. Let $\overline{E}$ be the set of edges that are not in E but whose complements are in E whose values we just added to x. Then all the cycles in $\overline{E}$ must also have total value at most $|C| - 1$. If this were not the case, then some cycle in E would have value less than 1, which means the solution x is not maximal. The last case we need to consider is a cycle composed of edges from both E and $\overline{E}$. Let A be the edges in this cycle from E and a denote their total value and let B be the set of edges in this cycle from $\overline{E}$ and b denote their total value. Assume $a + b > |C| - 1$ where A and B form cycle C. If we consider the complements of A and B then we have a cycle $\overline{C}$ with value $\overline{a} + \overline{b} < 1$.

The edges in $\overline{B}$ are in E and x is maximal. This means that there must be some cycle C' in E containing B such that the set of edges $C - B$ has total value $|C'| - 1 - \bar{b}$ and the set of its complementary edges has value $\bar{b}$. These complementary edges form a cycle with the edges in $\overline{A}$ which has value $\bar{a}$. Both sets are in $\overline{E}$ which implies that $\bar{a} + \bar{b} > 1$.

Therefore, all cycles in x have total value at most $|C| - 1$ and at least 1 so x is still a valid solution to LP_1. For each pair of vertices $i, j \in V$ such that (i, j) and (j, i) are not in E, the inductive step will be to show that we can find an assignment for x_{ij} and x_{ji} such that the resulting x is still a solution to LP_1 for the graph $G + (i, j)$. Thus, when we have assigned values to all pairs of vertices not associated with edges in E, we will have a solution for LP_1 for the complete graph. This must also be a valid solution to LP_2 since if every cycle has value at most $|C| - 1$ then every 2- and 3-cycle in the complete graph complies with the constraints in LP_2.

Now we prove the inductive step: to add an assignment to x we choose a pair of vertices i, j that are not connected by an edge in G. Let α be the length of the shortest path from i to j and β be the length of the shortest path from j to i. Together these shortest paths form a cycle in G. Therefore $\alpha + \beta$ is at least 1. We let edge (j, i) have value $x_{ji} = \max\{0, 1 - \alpha\}$ and edge (i, j) have value $x_{ij} = \min\{1, \alpha\} = 1 - x_{ji}$. If $\alpha > 1$, then any cycle that includes edge (j, i) has total value at least 1 and edge (i, j) has value 1 so any cycle that includes edge (i, j) has total value at least 1. If $\alpha \leq 1$, then any cycle that includes edge (j, i) will have total value at least 1 and every cycle that includes edge (i, j) will have total value at least $\alpha + \beta$, which is at least 1.

Since we can construct a solution for LP_2 with the same objective value as LP_1 so the integrality gap for LP_2 must be the same as LP_1. □

3.3 Constructing Undirected Dense Graphs with High Girth

In this section, we present a proof of a lemma that is due to Erdos and Sachs [10].

Lemma 11 *There exist undirected graphs with girth at least g and $\Theta(n \cdot (\frac{n}{8})^{\frac{1}{g}})$ edges.*

Proof. We will construct a graph with girth g and $\Theta(n \cdot (\frac{n}{8})^{\frac{1}{g}})$ edges. First, we construct a graph on n vertices by choosing d perfect matchings in $K_{\frac{n}{2},\frac{n}{2}}$ uniformly at random. Since each vertex has degree at most d, then for some vertex v there are at most d^g paths of length g that begin at this vertex. We will set $d^g = \frac{n}{8}$ (we will solve for d later). If we take a random walk of length g starting from vertex v, then the probability that we return to vertex v sometime during the walk is the number of vertices that we reach–at most d^g– divided by the total number of vertices in G. Since $d^g = \frac{n}{8}$, the probability that we return to vertex v after taking at most g steps is $= \frac{1}{8}$.

We will now use the Chernoff bound defined in Section 3.2. For each vertex, we can define an indicator random variable X_i. Let $X_i = 1$ denote that there is no cycle of length less than or equal to g that contains vertex i and let $X_i = 0$ denote that there is some cycle of length at most g that contains vertex i. Then $p_i = \frac{7}{8}$ for all i. X is the random variable representing the total number of vertices included in cycles of length less than or equal to g and $\mu = \frac{7n}{8}$. By Theorem 5, we have:

$$\Pr[X < (1-\epsilon)\tfrac{7n}{8}] \leq e^{\frac{\epsilon^2 \frac{7n}{8}}{2}}$$

For any fixed ϵ, this is very high probability, so we can assume that we have extremely close to $\frac{7n}{8}$ vertices that are not contained in cycles of length less than or equal to g. The remaining set of $\frac{n}{8}$ vertices may be contained in cycles of length less than or equal to g, but we can remove these vertices and all edges adjacent to them. This entails removing no more than $\frac{dn}{8}$ edges, so we are left with $\frac{dn}{2} - \frac{dn}{8} = \frac{3dn}{8}$ edges.

Now we solve for d: $d^g = \frac{n}{8}$, so $d = \frac{n}{8}^{\frac{1}{g}}$. Therefore $|E| = \frac{3n}{8}d = \frac{3n}{8}(\frac{n}{8}^{\frac{1}{g}})$. So we have found a graph with $\Theta(n \cdot (\frac{n}{8})^{\frac{1}{g}})$ edges and girth at least g. □

Chapter 4

Restricted Problem

In this chapter, we investigate the maximum acyclic subgraph problem restricted to certain classes of graphs, namely Eulerian graphs and graphs with maximum degree 3. Somewhat surprisingly, the general problem can be reduced to these special cases. First, we present an approximation-preserving reduction from the maximum acyclic subgraph problem in general graphs to the maximum acyclic subgraph problem restricted to Eulerian graphs. Then we use this reduction to give an approximation-preserving reduction from the maximum acyclic subgraph problem in general graphs to the maximum acyclic subgraph problem in graphs with maximum degree 3.

4.1 Eulerian Graphs

This reduction is due to Lázló Lovász and Fang Chen [8].

Theorem 6 *If for any $\delta > 0$, there exists a $(\frac{1}{2} + \delta)$-approximation algorithm for the maximum acyclic subgraph problem in Eulerian graphs, then there exists a $(\frac{1}{2} + \frac{\delta}{4})$-approximation algorithm for the maximum acyclic subgraph problem in general graphs.*

Proof. We will say a graph G is ϵ-far from Eulerian if there is a $v \in V$ for which $d^+(v)$ or $d^-(v) > (\frac{1}{2} + \epsilon)d(v)$. If $4\epsilon > \delta \Rightarrow \epsilon > \frac{\delta}{4}$, then we can make at least an ϵ (or $\frac{\delta}{4}$) gain at each vertex (by placing the greater of the in and out edges in the acyclic subgraph) until the graph is less the ϵ-far from Eulerian.

If $4\epsilon < \delta \Rightarrow \epsilon < \frac{\delta}{4}$, we will add a new vertex v^* to G to obtain a new graph $G + v^*$. To the vertex v^* we will attach in edges and out edges from and to each vertex in G for which $|d^+(v) - d^-(v)| > 0$, thus making $G + v^*$ Eulerian. Let $OPT(G)$ denote the size of the maximum acyclic subgraph of G. Then:

$$d(v^*) = \sum_{v \in G} |d^+(v) - d^-(v)| \leq \sum_{v \in G} 2\epsilon d(v) \leq 2\epsilon \sum_{v \in G} d(v) \leq 4\epsilon |E| \leq 4\epsilon OPT(G)$$

If we have a $(\frac{1}{2} + \delta)$–approximation for $G + v^*$, then:

$$\begin{gathered} OPT(G + v^*) \geq OPT(G) + \tfrac{1}{2} d(v^*) \;\; \Rightarrow OPT(G + v^*) - d(v^*) \;\; \geq \\ (\tfrac{1}{2} + \delta) OPT(G + v^*) - d(v^*) \geq \\ (\tfrac{1}{2} + \delta) OPT(G) + (\tfrac{1}{2} + \delta) \tfrac{1}{2} d(v^*) - d(v^*) \;\; \geq \;\; (\tfrac{1}{2} + \delta) OPT(G) - \tfrac{3}{4} d(v^*) \geq \\ (\tfrac{1}{2} + \delta) OPT(G) - 3\epsilon OPT(G) \;\; \geq \;\; (\tfrac{1}{2} + \delta - 3\epsilon) OPT(G) \end{gathered}$$

Since $4\epsilon < \delta \Rightarrow \epsilon < \frac{\delta}{4}$:

$$(\frac{1}{2} + \delta - 3\epsilon) > (\frac{1}{2} + \delta - \frac{3}{4}\delta) = (\frac{1}{2} + \frac{\delta}{4})$$

So we can get a $(\frac{1}{2} + \frac{\delta}{4})$-approximation algorithm for all graphs given a $(\frac{1}{2} + \delta)$-approximation algorithm for Eulerian graphs. □

4.2 Degree-3 Graphs

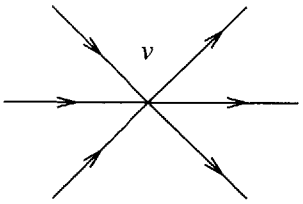
The maximum acyclic subgraph problem remains NP-hard even for graphs with maximum degree 3 [7]. (For graphs with maximum degree 2, the problem is easy.) In this section, we give an approximation-preserving reduction from the maximum acyclic subgraph problem for Eulerian graphs to the maximum acyclic subgraph problem for graphs with maximum degree 3.

Theorem 7 *If for any $\epsilon > 0$, there exists a $(\frac{17}{18} + \epsilon)$-approximation algorithm for the maximum acyclic subgraph problem in graphs with maximum degree 3, then there exists some $\delta > 0$ such that there is a $(\frac{1}{2} + \delta)$-approximation algorithm for the maximum acyclic subgraph problem in general graphs.*

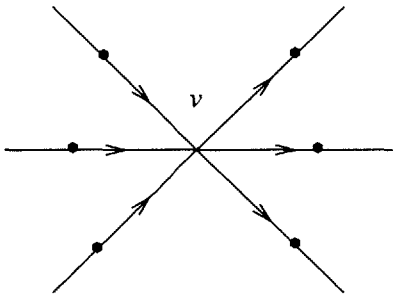
To prove this theorem we will first introduce the following lemmas.

Lemma 12 *Given an Eulerian graph $G = (E_G, V_G)$ we can construct a 3-regular graph $G' = (E_{G'}, V_{G'})$ with $|E_{G'}| = 9|E_G| - 9|V_G|$ such that the size of the minimum feedback arc set in G is the same size as the minimum feedback arc set in G'.*

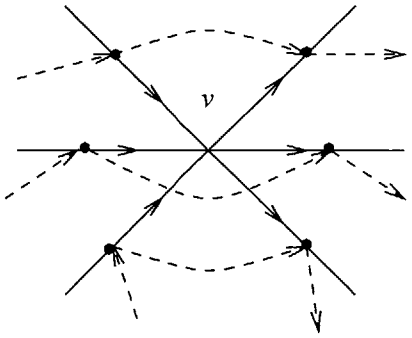
Proof. Since G is Eulerian, we will use $d(v)$ to denote both the in- and out-degree of vertex v for this proof. The figure below shows a vertex v in G before we add any vertices or edges.



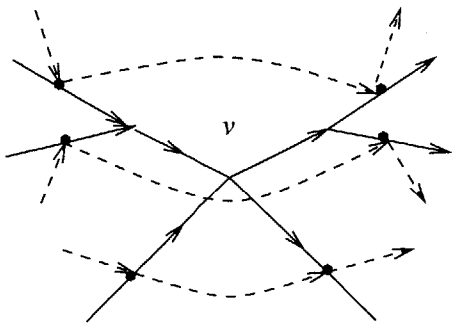
The construction of G' is as follows: we first place a vertex in the middle of each edge of G.



The number of vertices is now $|V_G| + |E_G|$. Then for each vertex $v \in V_G$, we match each incoming edge with a distinct outgoing edge by adding an edge from the vertex placed on the incoming edge to the vertex placed on the outgoing edge. (Since G is Eulerian, we will always be able to find such a matching.)

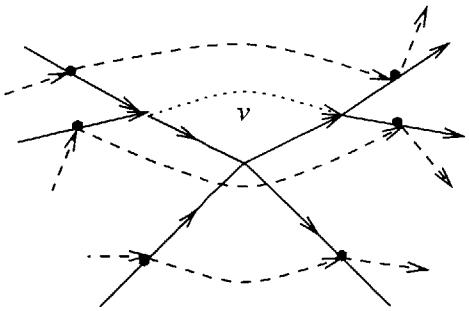


All of the $|E_G|$ new vertices now have in- and out-degree 2. We want to replace the vertices from V_G with a new set of vertices in which each vertex has in- and out-degree 2 so that the entire graph will be 4-regular and Eulerian. Consider vertex $v \in V_G$. We build a binary tree from v to the $d(v)$ new vertices placed on the outgoing edges.



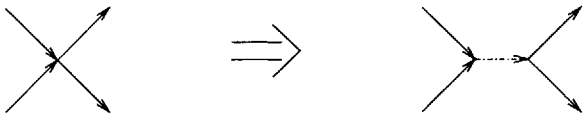
This requires $d(v) - 2$ new vertices. We can see this by the following reasoning: let p be the highest power of 2 not greater than $d(v)$. For a binary tree that connects p vertices to v, we need $\frac{p}{2} + \frac{p}{4} + \ldots + 2 = d(v) - 2$ new vertices or vertices that are internal nodes on the binary tree. For each of the remaining $d(v) - p$ vertices, we will connect two vertices to one of the p leaves thus adding just one internal vertex for each of these vertices. Therefore, we have a total of $d(v) - p + p - 2 = d(v) - 2$ internal or new vertices on the binary tree.

We also build a binary tree from v to the $d(v)$ new vertices placed on the incoming edges. Then we match each vertex from the incoming binary tree with a distinct vertex from the outgoing binary tree by adding an edge from the former to the latter.



We build these two binary trees and match their vertices as we just described for all vertices $v \in V_G$. The total number of new vertices needed to build these two binary trees for vertex v is $2d(v) - 4$. The total number of new vertices needed to build two binary trees for all $v \in V_G$ is therefore $\sum_{v \in V_G} 2d(v) - 4 = 2|E_G| - 4|V_G|$.

We now have a 4-regular Eulerian graph with $(|E_G| + |V_G|) + (2|E_G| - 4|V_G|) = 3|E_G| - 3|V_G|$ vertices. We are not yet finished constructing G' since we want G' to be a 3–regular graph, but we will call this intermediate graph G''. The last step in the construction of G' is to stretch each vertex in the current graph into an edge so that G' is 3–regular.



After the final step in the construction of G', $|E_{G'}| = 3(3|E_G| - 3|V_G|) = 9|E_G| - 9|V_G|$.

We will use the following definition later on in the proof.

Definition 1 *A* blue *edge is an edge (i, j) such that vertex i has in-degree 2 and vertex j has out-degree 2.*

Note that the edges in G' that correspond to vertices in G'' are blue edges.

We will now show that this reduction preserves the size of minimum feedback arc set, i.e. the size of the minimum feedback arc set of G' is equal to the size of the minimum

feedback arc set of G. Specifically, we will show that given a feedback arc set in G, we can construct a feedback arc set in G' of the same size. Conversely, given a feedback arc set in G', we can construct a feedback arc set in G of size at most the size of the feedback arc set in G'.

(i) Suppose F is a feedback arc set of G. We will construct F', a feedback arc set of G'. For each edge $e_{ij} \in F$, we add to F' the blue edge from $E_{G'}$ that corresponds to the vertex used to subdivide edge e_{ij} in the first stage of the construction of G'. Then $|F'| = |F|$ and $E_{G'} - F'$ is acyclic by the following proof.

Assume $E_{G'} - F'$ is not acyclic. Note that every blue edge corresponds to an original edge, to an original vertex, or to a vertex on a binary tree in G''. If we take a walk on the graph G' starting from a blue edge corresponding to an edge e_{ij} in G (i.e. to a vertex used to subdivide an edge in the first step of the construction) we will either follow a path on the binary tree leading to a blue edge corresponding to edge e_{jk} or we will walk directly to edge e_{jk} for some k. Therefore, if we find a cycle in G', the set of edges in that cycle that correspond to edges in E_G are still present in G implying that there is a cycle in G, which is a contradiction to the fact that F is a feedback arc set.

(ii) Suppose F' is a feedback arc set in G'. Assume all edges in F' are blue edges. If they are not, we can replace them with blue edges adjacent to the non-blue edges and obtain a feedback arc set of equal or smaller size. Then for every edge in F' that corresponds to an edge in E_G, add the corresponding edge in E_G to F. Then $E_G - F$ is acyclic by the following proof.

Assume $E_G - F$ is not acyclic. Then consider some cycle in $E_G - F$. The blue edges corresponding to each edge e_{ij} in the cycle are still in $E_{G'} - F'$. Since $E_{G'} - F'$ is acyclic, at least one of the edges used to connect vertices that we placed in the middle of the edges in E_G must have been removed. But this is a contradiction, because these edges are not blue edges, and we converted F' to a feedback arc set that contained only blue edges. □

Corollary 13 *A maximum acyclic subgraph in G of size S corresponds to a maximum acyclic subgraph in G' of size $S + 8|E_G| - 9|V_G|$.*

Proof. Given a maximum acyclic subgraph in G of size S, we can find a feedback arc set in G of size $E_G - S$. Therefore, we can find a feedback arc set in G' of size $E_G - S$. There will be $S + 8|E_G| - 9|V_G|$ edges remaining in $E_{G'}$ and these edges make up an acyclic subgraph of G'. □

Lemma 14 *We can convert an acyclic subgraph of G' of size at least $(\frac{17}{18} + \epsilon)MAS(G')$ to an acyclic subgraph of G of size at least $(\frac{1}{2} + \delta)MAS(G)$ for some constants $\epsilon, \delta > 0$.*

Proof. Let $MAS(G)$ denote the size of the maximum acyclic subgraph of G. We assume that $MAS(G)$ is at least βE_G for some fixed $\beta < 1$ (we will explain β later). If $MAS(G)$ is less than βE_G, then we can find an acyclic subgraph in G with at least half the edges of G thereby obtaining a $(\frac{1}{2} + \epsilon)$-approximation for some $\epsilon > 0$.

Say we are given an α-approximation algorithm for 3-regular graphs. We can take an Eulerian graph G and convert it a 3-regular graph G' using the construction described previously. Then we can find an acyclic subgraph S' for G' that is of size at least $\alpha MAS(G')$. By Corollary 13 we have:

$$\alpha MAS(G') = \alpha(MAS(G) + 8E_G - 9V_G)$$

To find an acyclic subgraph S of G given S', we remove all edges from S' that do not correspond to blue edges representing original edges of G. There are at most $8E_G - 9V_G$ such edges, since E_G of the edges in G' correspond to edges in G. So when we remove these edges from S', we are left with a set S of size at least:

$$\alpha(MAS(G) + 8E_G - 9V_G) - (8E_G - 9V_G) = \alpha MAS(G) + (8\alpha - 8)E_G + (9 - 9\alpha)V_G$$

Since $E_G \leq \frac{MAS(G)}{\beta}$, $8\alpha - 8 < 0$, and $V_G \geq \frac{E_G}{d} > \frac{MAS(G)}{d}$, where d is the average degree of G, we have:

$$\alpha MAS(G) + (8\alpha - 8)E_G + (9 - 9\alpha)V_G \geq \alpha MAS(G) + (8\alpha - 8)\frac{MAS(G)}{\beta} + (9 - 9\alpha)\frac{MAS(G)}{d}$$

And:

$$\alpha + \tfrac{8\alpha}{\beta} - \tfrac{8}{\beta} + \tfrac{9}{d} - \tfrac{9\alpha}{d} > \alpha + \tfrac{8\alpha}{\beta} - \tfrac{8}{\beta} > \tfrac{1}{2} \Rightarrow \alpha + \tfrac{8\alpha}{\beta} > \tfrac{1}{2} + \tfrac{8}{\beta}$$

Since $\beta < 1$:

$$\alpha > \tfrac{(\beta+16)}{(\beta+8)}\tfrac{1}{2} > \tfrac{17}{18}$$

Therefore, if we found a $(\frac{17}{18}+\delta)$-approximation for 3-regular graphs, then we could find some β such that the above inequality is true. If $MAS(G) \geq \beta E_G$, then we could use the reduction and the $(\frac{17}{18}+\delta)$-approximation algorithm for degree-3 graphs to find a $(\frac{1}{2}+\epsilon)$–approximation for Eulerian graphs, which would by Theorem 6 lead to a $(\frac{1}{2} + \frac{\epsilon}{4})$-approximation for general graphs. □

Theorem 8 *There is no PTAS for the maximum acyclic subgraph in degree-3 graphs unless* P = NP.

Proof. If we have an α-approximation for the maximum acyclic subgraph in degree-3 graphs, then we will get a $(\alpha + \frac{8(\alpha-1)}{\beta})$-approximation for Eulerian graphs. Therefore, if $\alpha = 1 - \epsilon$ and if $\beta \geq \frac{1}{2}$, then:

$$\alpha + \tfrac{8(\alpha-1)}{\beta} = 1 - \epsilon - \tfrac{8\epsilon}{\beta} \geq 1 - 17\epsilon$$

Thus, if we can find a $(1-\epsilon)$–approximation for the maximum acyclic subgraph in degree-3 graphs, then we can find a $(1-\delta)$-approximation for Eulerian graphs, where $\delta = 17\epsilon$, which implies that we can find a PTAS for Eulerian Graphs. This implies that we can find a PTAS for general graphs, which is a contradiction, since the maximum acyclic subgraph problem for general graphs does not admit a PTAS. (See Chapter 2 for a proof of this.) □

Chapter 5

Algorithms for Degree-3 Graphs

In [1] an algorithm that returns an acyclic subgraph of size $\frac{2}{3}|E|$ is given for graphs with maximum degree 3 and an algorithm that returns an acyclic subgraph of size $\frac{13}{18}|E|$ is given for 3-regular graphs. In this chapter, we show that the maximum acyclic subgraph problem in graphs with maximum degree 3 can be approximated to within $\frac{8}{9}$ of optimal using simple combinatorial methods.

5.1 Assumptions

Given a graph G with maximum degree 3 for which we want to find an acyclic subgraph S, we can assume the following in any algorithm:

(i) All vertices have total degree exactly 3. If there is any vertex v such that $d^+(v) = d^-(v) = 1$, then we can remove the vertex and treat the adjacent edges as one edge. Thus an edge in the modified graph may represent a path in the original graph. If we ever add an edge representing a path to S, then we are really adding all the edges in the path to S. If we do not add this edge to S, then we can remove any one edge from the path and add the rest of the edges on the path to S. If for any vertex $d^+(v) = 0$ or $d^-(v) = 0$, we can add all edges adjacent to v to S.

(ii) G has no 2- or 3-cycles, since we can treat both optimally. For 2-cycles, consider the 2 adjacent non-cycle edges. If they are both in edges, or both out edges, then we can break the two cycle by removing an arbitrary edge. If one is out and the other is in, then one of the edges in the 2-cycle is consistent with the direction of a possible cycle containing both of the two non-cycle edges, so we can remove this edge. For 3-cycles, if we contracted the 3-cycle, we would get a new vertex of degree 3. If this vertex is a source or a sink, then we can remove an arbitrary edge from the 3-cycle. Otherwise, we remove an edge from the 3-cycle, so that the path to or from the single in or out edge is broken.

5.2 Algorithm 1

We will now consider the case where G has no blue edges. See Section 4.2 for the definition of a blue edge. If there are no blue edges, then we can find the maximum acyclic subgraph in polynomial time. For our algorithm, we will use the following lemma:

Lemma 15 *If G has maximum degree 3 and contains no blue edges, then all cycles in G are edge disjoint.*

Proof. Assume that there are some 2 cycles in G that have an edge (or a path) in common. First case: assume that these two cycles have an isolated edge (i, j) in common, i.e. edge (i, j) belongs to both cycles, but edges (a, i) and (j, b) each belong to only one of these cycles. Then vertex i must have in-degree 2 and vertex j must have out-degree 2. Thus, edge (i, j) is a blue edge, which is a contradiction. Second case: assume these two cycles have a path $\{i, \cdots, j\}$ and that this path is maximal, i.e. edge (a, i) and (j, b) each belong to only one of these cycles. Vertex i must have in-degree 2 and vertex j must have out-degree 2. Therefore, one of the edges on the path must be a blue edge, which is a contradiction. □.

Since all the cycles in a graph with no blue edges are edge disjoint, we can find the maximum acyclic subgraph of such a graph in polynomial time. Given a graph G containing no blue edges, the following is an algorithm to find the maximum acyclic subgraph of G:

```
Algorithm 1:

    Step 0. Let S={},G'=G.
    Step 1. While G' is not acyclic, do:
            Step 1a. Find a cycle in G'.
            Step 1b. Remove an edge in this cycle from G'.
            Step 1c. Remove the rest of the edges in this cycle from G' and
                     add them to S.
    Step 2. Add the remaining edges to S.
    Step 3. Output S.
```

5.3 Algorithm 2

If G has blue edges, then the problem is NP-hard. For this case, we will give the following $\frac{8}{9}$-approximation algorithm. See Section 5.1 for an explanation of Step 1a.

```
Algorithm 2:

    Step 0. Let S={}, G'=G.
    Step 1. While there are still blue edges in G', do:
            Step 1a. Optimally treat any 2- and 3-cycles.
            Step 1b. Find a blue edge e in G'.
            Step 1c. If e is contained in a component with exactly 9
                     edges then:
                           Solve for the maximum acyclic subgraph of
                           this component exactly.
                     Else: Remove the four edges neighboring e from G'
                           and add them to S.
            Step 1d. Contract any vertices with in-degree and out-degree 1.
    Step 2. If G' is acyclic, then add all edges in G' to S.  If G' is not
            acyclic, then since it contains no blue edges, use
            Algorithm 1 to find the maximum acyclic subgraph of G' and
            add it to S.
    Step 3. Uncontract every edge added to S that was contracted in Step
            4.  For every contracted edge that was removed from G' and
            not added to S, remove any edge from the corresponding path
            in G and add the remaining edges to S.
    Step 4. Output S.
```

Theorem 9 *Algorithm 2 is an $\frac{8}{9}$-approximation for the maximum acyclic subgraph problem.*

Proof. We will show that for each iteration of Step 1 through Step 5 of the algorithm, for every edge we remove, we contract or add to S a total of at least 8 edges. Edges that we contract will be added to S in Step 3.

Consider a blue edge (i, j) in G. There must be four distinct vertices within distance 1 from i and j (since there are no 2- or 3-cycles). So there are 6 vertices that are at no more than one edge away from vertex i or vertex j (including vertices i and j). Therefore, there must be at least $\frac{3\times 6}{2} = 9$ edges in this neighborhood. If there are exactly 9 edges, then we have found a connected component and Algorithm 2 will solve this component exactly. Otherwise, if there are more than 9 edges in the neighborhood of edge (i, j) (i.e. there could be as many as 12 edges) then for each of the 4 distinct vertices that are exactly one edge away from i or j, we can either contract this vertex, or we can add two more edges to S (which would let us add more than 8 edges to S in this round). Thus, for every one

edge we remove from G, we add at least 8 edges to S which proves that Algorithm 2 is a $\frac{8}{9}$-approximation algorithm. □

Bibliography

[1] Bonnie Berger and Peter W. Shor. Tight Bound of the Maximum Acyclic Subgraph Problem. In *Journal of Algorithms*, 25, pages 1–18, 1997.

[2] M. X. Goemans and L. A. Hall. The Strongest Facets of the Acyclic Subgraph Polytope are Unknown. In *Proceedings of IPCO 1996*, LNCS Vol 1084, pages 415–429.

[3] M. Grötschel, M. Jünger, and G. Reinalt. On the Maximum Acyclic Subgraph Polytope. In *Mathematical Programming*, 33, pages 28–42, 1985.

[4] M. Grötschel, M. Jünger, and G. Reinalt. Facets of the Linear Ordering Polytope. In *Mathematical Programming*, 33, pages 43–60, 1985.

[5] M. Grötschel, L. Lovász, and A. Schrijver. Geometric Algorithms and Combinatorial Optimization Springer, 1988.

[6] Johan Håstad. Some Optimal Inapproximability Results. In *Proceedings of STOC*, 1997.

[7] Richard M. Karp. Reducibility Among Combinatorial Problems. In *Complexity of Computer Computations*, Plenum Press, 1972, pages 85-104.

[8] Lázló Lovász and Fang Chen. Personal communication.

[9] Rajeev Motwani and Prabhakar Raghavan. Randomized Algorithms. Cambridge University Press, 1995.

[10] Dan Spielman. Applied Extremal Combinatorics Lecture Notes, Lecture 9, Fall 1998. Lecture notes available at: www-math.mit.edu/~spielman/AEC/notes.html.

[11] Vijay V. Vazirani. Approximation Algorithms. Preliminary draft of book available at: www.cs.gatech.edu/fac/Vijay.Vazirani.