THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GREEDY ALGORITHM FOR APPROXIMATION OF THE MAXIMUM INDUCED MATCHING PROBLEM IN REGULAR GRAPHS OF GIRTH AT LEAST SIX

CHAITANYA PATEL
SPRING 2016

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Computer Engineering and Mathematics
with honors in Computer Science

Reviewed and approved* by the following:

Dr. Piotr Berman
Associate Professor of Computer Science
Thesis Supervisor

Dr. John Hannan
Head of Department of Computer Science
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

An induced matching of a graph G is the edge set of an induced subgraph of G, which is also a matching, i.e., its edge are disjoint. The problem that is discussed in this thesis is the Maximum Induced Matching (MIM for short) problem and its aim is to maximize the size of the matching. The problem has been modeled as the Set Packing and the Maximum Independent Set problem. These problems are very difficult to approximate and for this reason there was extensive research on restricted classes of graphs. The work done in this thesis further improves on a simple linear time greedy algorithm that gives an approximation ratio of $d - 1$ for a d-regular graph. This approximation ratio was improved to $\frac{5}{3}$ for $d = 3$. In this thesis, we further improve this ratio to be $\frac{3}{2}$ from the previous $\frac{5}{3}$ for $d = 3$ (3-regular graphs) under the assumption that the graph has no cycles of length 5 or less. Moreover, we conjecture that by following a similar methodology we can generalize the result with the approximation ratio being $d/2$ for a d-regular graph without cycles of length smaller than 6.

Table of Contents

List of Figures

Acknowledgements

I wish to express my sincere thanks to Dr. Piotr Berman, my thesis supervisor at The Pennsylvania State University, without whose help I would not have completed the thesis to this degree of satisfaction.

I am grateful to the Schreyer Honors College at The Pennsylvania State University for giving me the opportunity to work on such a project. I have definitely learned a lot from this invaluable experience.

I thank Dr. John Hannan, my honors adviser, and the Computer Science and Engineering department at The Pennsylvania State University for all their help and support I received during my undergraduate studies.

Lastly, I would like to thank my parents, Mr. Nitin Patel and Mrs. Shradhdha Patel for encouraging me to excel throughout my academic endeavors and supporting me at all times.

Chapter 1

Introduction

1.1 Maximum Independent Set problem

The maximum independent set problem (MIS for short) is one of the most important problems in graph theory and in the theory of algorithms. An undirected graph $G = (V, E)$ has node set V and edge set E where each edge $e \in E$ is an unordered pair of nodes from V. A subset of nodes $U \subset V$ defines its induced set of edges $E[U] = \{e \in E : \ e \subset U\}$, where the statement $e \subset U$ means that the nodes of the edge e make a subset of U. Set U is *independent* if and only if $E[U] = \emptyset$. The MIS problem is to find the maximum size of an independent set in the input graph.

The concept of the MIS allows to model a very large number of problems of practical importance. Unfortunately, very soon after defining concept of NP-completeness it was established that the decision problem: does the input graph contain an independent set of size k? is NP-complete and it belongs to so-called intractable problems [1].

However, since MIS problem is so important, many additional results were proven. A series of algorithms solve MIS problem exactly; naively we would inspect 2^n sets as candidates for the solution (where $n = |V|$), but Tarjan and Trojanowski in 1976 [2] showed an algorithm that achieves the same goal in time $O(2^{n/3})$; in 1986 this was further improved to $O(2^{0.304})$ by Tang Jian [3] who at that time was a student of Penn State.

1.1.1 MIS is hard to approximate

Another direction was to address the approximation of the problem: given that the size of optimal solutions is m, can we find in polynomial time solutions of size at least $m/2$? Or perhaps if that is too difficult, of size $m/100$? Or if that is too difficult, of size $\sqrt{m}$? All these questions were answered negatively by Håstad in 1999 [4]. In that paper the main result was that for every

positive $\epsilon < \frac{1}{2}$ one can prepare two families of graphs, *BigSet* and *SmallSet*, so that the size of maximum independent sets is at least $n^{1-\epsilon}$ in the former family, and smaller than n^{ϵ} in the latter. Astonishingly, when given a graph G from *BigSet*$\cup$*SmallSet*, it is NP-hard to decide if $G \in$ *BigSet*.

This famous result allows to show that for every k it is NP-hard to produce solutions for MIS instances in such a way that if the optimum solution has size m, we will find a k-good solutions in the sense that it has size at least $\sqrt[k]{m}$. With this goal in mind we use $\epsilon = 1/(k+1)$. Suppose that we have a polynomial time algorithm that produces k-good solutions. Then for a graph from *BigSet* there exists a solution of size $m \geq n^{1-1/(k+1)} = n^{k/(k+1)}$, and the k-good solution found by our algorithm has size at least $\sqrt[k]{n^{k/(k+1)}} = n^{1/(k+1)} = n^{\epsilon}$. This solution would prove that the graph does not belong to *SmallSet*, contradicting the theorem of Håstad (assuming that P $\neq$ NP).

1.2 Set Packing

In the light of Håstad result, there is no hope of finding practically useful solutions to MIS for graphs with more than, say, 150 nodes. However, the reason MIS is so important is that it models many optimization problems that arise in practice. One family of such models uses the form of MIS called *Set Packing*. We are given a collection of objects R, intuitively, resources, and a collection of subsets $\mathcal{S} = \{S_1, \ldots, S_n\}$, intuitively, possible projects. Our aim is to complete as many projects as possible, but each resource can be used only for one project. In other words, a solution is a subset $\mathcal{P} \subset \mathcal{S}$ such that for every $S_i, S_j \in \mathcal{P}$ we have $S_i \cap S_j = \emptyset$/ In terms of MIS we can form a graph with $V = \mathcal{S}$ and edge of the form $\{S_i, S_j)$ such that $S_i \cap S_j \neq \emptyset$. So far this is not very helpful because there is no big difference between MIS and maximum set packing.

To see that "there is no difference" we will translate an instance of MIS, say (V, E) into an instance of set packing. We will have set $R = E$, and a for each $u \in V$ we define set $S_u = \{e \in E : \ u \in e\}$, then we define $\mathcal{S} = \{S_u : \ u \in V\}$. One can see see that $S_u \cap S_v \neq \emptyset$ if and only if $\{u, v\} \in E$.

1.2.1 Approximating of Maximum Set Packing is easy is sets are small

Even with the negative result we have just discussed, the framework of set packing is helpful because it allows to formulate a restricted version of Set Packing that we may call d-Set Packing. In this problem we allow only instances in which sets of $\mathcal{S}$ have at most d elements.

Note that 2-Set Packing is the same as Maximum Matching, and thus it has a polynomial time algorithm. But for $d > 2$ this problem is NP-complete, in particular for $d = 3$ this is 3-Dimensional Matching, and NP-complete problem described in [1]. However, a very simple greedy algorithm provides so-called factor d approximations, i.e. it finds solutions of size at least opt/d, where opt is the size of the optimal solution of the problem. This algorithms is as follows: select ANY set S from $\mathcal{S}$ and form the residual instance by removing from $\mathcal{S}$ every set that contains an element of S. We stop that process when the residual instance contains no sets.

To see that this algorithm obtains at least opt/d sets, we consider an optimum solution $\mathcal{P}$. In the analysis, when the algorithm computes the residual instance we obtain the residual solution by removing from $\mathcal{P}$ every set that contains an element of S. Because $|S| \leq d$ and sets in $\mathcal{P}$ are disjoint, a single step of finding the residual solution reduces the number of sets in $\mathcal{P}$ by at most d.

Local improvements algorithm modify the current solution to a combinatorial problem by replacing a small set of selected object with a a larger set. It was shown that for every d there exists a polynomial time algorithm for approximating d-Set Packing that achieves the ratio *opt/achieved* which is at most $\frac{d+2}{3}$ [5].

1.3 Maximum Induced Matching − a special set packing problem (previous work)

The graphs considered in this thesis are undirected and simple − the edge set is a set of unordered pairs of nodes (2-element sets of nodes). Given a graph $G = (V, E)$ and a node set $U \subset V$ we define $G[U]$, the *induced subgraph* of G as $(U, E[U]$ where $E[U]$ consists of all edges of E that are contained in U. A *matching* is a set of edges that are pairwise disjoint. An *induced matching* is an edge set M that is a matching and also an edge set of an induced subgraph. One can see that a matching M is an induced matching of (V, E) if and only if there are no edges in E that intersect two different edges of M.

This automatically allows to model maximum induced matching as a special case of set packing. With every edge $e \in E$ we associate the set of edges that intersect it,

$$\textit{DoubleStar}(e) = \{e' \in E : \ e \cap e' \neq \emptyset\}. \tag{1.1}$$

Then $M \subset E$ is an induced matching if and only if $\textit{DoubleStar}(e) \cap \textit{DoubleStar}(e')$ for every $e, e' \in M$. For this reason, an induced matching is equivalent to a set packing of DoubleStars.

A d-regular graph is a simple undirected graph such that each node belongs to d edges, in other words, each nodes has exactly d neighbors. In a d-regular graph each DoubleStar consists of $2d - 1$ edges. Therefore if we want to offer algorithms for Maximum Induced Matching problem we need to check the implications of the results on d-Set Packing, more precisely, for $(2d - 1)$-Set Packing. One could also observe that if (1.1) is satisfied then $\textit{DoubleStar}(e)$ and $\textit{DoubleStar}(e')$ share an edge that is neither e not e', and thus Maximum Induced Matching in d-regular graph can be modeled with $(2d - 2)$-Set packing.

For that reason the approximation ratio of $2d-2$ is trivially achievable with the simplest greedy algorithm and $\frac{2d}{3}$ is achievable with much slower local search.

Duckworth *et al.* [6] showed a greedy algorithm that achieves ratio $d - 1$, thus considerably better than a greedy algorithm for $(2d - 1)$-Set Packing, although not as good as the local search based on "large sets". (It must be stressed that local search is considerably slower than a greedy algorithm, and "large sets" local search is exclusively theoretical.)

Li [7] found that the greedy algorithm of Duckworth *al.* can be improved to give ratio $5/3$ for $d = 3$, which is better than 2, and they conjectured that a similar improvement is possible for every $d > 2$.

1.4 Our results

The extension of the result of Li [7] to d-regular graphs with $d > 3$ is made difficult by the fact that their technique catalogs all situations posed by short cycles in the graph, namely the cycles

of five or four nodes. This raised the following question: is the problem measurably easier if the graph does not have such cycles?

Here we answer this question affirmatively. For 3-regular graphs a simple greedy algorithm guarantees the approximation of $3/2$ which is better than the previously achieved $5/3$. We also conjecture that a similar algorithm can yield $d/2$ approximation for arbitrary d.

More formally, girth of the graph is the length of its shortest cycle. We exhibit a greedy linear time algorithm for 3-regular graphs with girth 6 or larger that has approximation ratio $3/2$.

1.5 Further work

As we will mention in the Conclusions, we plan to generalize the results of this thesis to d-regular graphs for every d.

Another challenge is is worth to consider is to extend the results to a wider class of graphs.

In d-Set Packing framework, the problem of maximizing the size of an induced matching is equally easy if instead of d-regular graphs we consider graphs of degree d, i.e. if we allow nodes to have d neighbors or less than d. This stems from the fact that the size of DoubleStars at most $2d - 1$ in graphs of degree d. But the results of Duckworth et al [6] and Li [7] and those in this thesis apply only to d-regular graphs because they utilize the trivial upper bound that is available for d-Set Packing: if we want to pack sets of size at least d that are contained in a universal set of size n then we can pack at most n/d sets.

This works very well for Induced Matching in d-regular graphs, but in a wider class of graphs of degree d some DoubleStars can have fewer than $2d-1$ edges. Because the proven approximation ratio has the form "guaranteed size delivered by our algorithm / "upper bound", this generalization would require to connect the potential defined to analyze our algorithm with an upper bound that would be applicable in graphs of degree 3 (and other fixed degrees).

Chapter 2

3-regular graphs – Preliminaries

2.1 Basic Terminology and Upper and Lower Bounds

Before we begin analyzing the problem, we define basic terms used that may not be intuitive.

Definition The $degree$ of a node is defined to be the number of edges that contain the node. In other words, it is the number of nodes that this node is connected to via edges.

Definition A d-regular graph is a graph such that every node has a degree d.

Definition A set U_i, where i is an integer greater than 0, is the set of all nodes in the current graph that have a degree i. For example, when we say U_2, we refer to the set of all nodes whose degree is 2 in the graph.

For the problem, it is necessary that we define the upper and lower bounds of the problem to create the approximation ratio. The approximation ratio is the ratio of the size of optimal matching to the size of the matching produced by our algorithm.

In [6] it was shown that the size of an induced matching in a d-regular graph $G(V, E)$ is at most $\lfloor \frac{|E|}{2d-1} \rfloor$. For d = 3 we have $|E| = \frac{3}{2}|V|$, hence this upper bound is $\lfloor \frac{|E|}{5} \rfloor = \lfloor \frac{3|V|}{10} \rfloor$.

Our algorithm finds an induced matching of size at least $\lceil \frac{|V|-1}{5} \rceil$, which guarantees that its size is at least $\frac{2}{3}$ of the optimum result as explained by following lemma:

Lemma 2.1.1 *For every positive even integer greater than 0, n, we have $\lceil \frac{n-1}{5} \rceil \geq \frac{2}{3} \lfloor \frac{3n}{10} \rfloor$.*

Proof. We will prove this by mathematical induction.
Base Cases:

For $n = 2, 1 \geq 0$.
For $n = 4, 1 \geq \frac{2}{3}$.
For $n = 6, 1 \geq \frac{2}{3}$.
For $n = 8, 2 \geq \frac{4}{3}$.
For $n = 10, 2 \geq 2$.

Inductive Step: We will now prove that the claim holds for n, assuming it holds for $n - 10$. We have $\lceil \frac{n-1}{5} \rceil = (\lceil \frac{(n-10)-1}{5} \rceil + 2) \geq (\frac{2}{3} \lfloor \frac{3(n-10)}{10} \rfloor + 2) = (\frac{2}{3} \lfloor \frac{3(n-10)}{10} + 30 \rfloor) = \frac{2}{3} \lfloor \frac{3n}{10} \rfloor$. The inequality here holds true by the inductive hypothesis. ❒

With the lemma above, we have defined the upper and lower bounds of the number of edge selections in the induced matching problem. Thus, our algorithm will aim for selecting at least $\lceil \frac{|V|-1}{5} \rceil$ edges, where V is the set of vertices in the input graph $G(V, E)$.

Chapter 3

3-regular graphs – Greedy algorithm

A greedy graph algorithm is described by two rules:

1. A rule to define a residual graph after a selection is made, and
2. A rule to make a selection given an input graph

In our case, we use rule (2) to select an edge for our induced matching and then rule (1) to form the residual graph. After creating the residual graph, we treat this residual graph as the input graph for the next selection until we can no longer make a selection, i.e., until the residual graph is empty.

The following two sections will describe these two rules that will make up the greedy algorithm.

3.1 Greedy algorithm part 1 – defining the residual graph after an edge selection

Using the input graph $G(V, E)$, our greedy algorithm starts with $U = V$ and makes an edge selection in the induced subgraph, $G[U]$, where U is the set of all nodes that are not eliminated from the graph. Elimination of nodes is caused by selecting edges. In particular, after selecting edge e the algorithm eliminates all the nodes that cannot belong to edges selected afterwards. Therefore, nodes are eliminated by the following rules:

(a) nodes of edge e;

(b) nodes adjacent to e;

(c) nodes with no neighbors in U after elimination of types (a) and (b) above.

The maximum induced matching problem is easier to approximate in graphs of at least six because of the following property:

Lemma 3.1.1 *The node eliminations associated with an edge selection can decrease the degree of a node (the number of neighbors in U) by at most 1.*

Proof. Suppose selecting edge e eliminates two neighbors of node b, say c and d, while b remains in U. Then c and d were eliminated by rule (b). Thus, they have neighbors in edge e. This implies the following cycle:

from b to c to a node of e (optionally, to the other node of edge e) to d to b.

This cycle has length 4 or 5 which contradicts the assumption that the girth of the graph $G(V, E)$ exceeds 5 (See Figure 3.1).

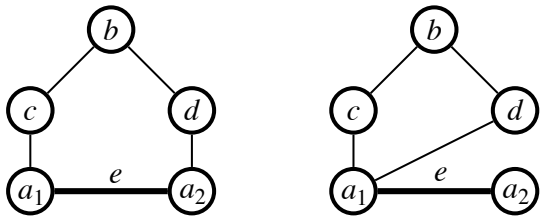

Figure 3.1: Figures to show cycles of length 4 or length 5

❒

Intuitively, we wish to assure that the first edge selection removes 6 nodes and each subsequent edge selection removes at most 5 nodes. However, this is not true. This is shown in Figure 3.2.

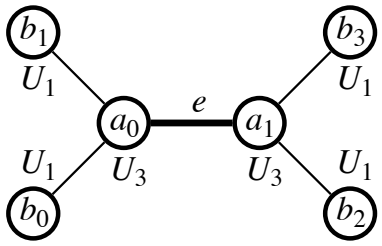

Figure 3.2: The elimination of 6 nodes after selecting edge e.

Instead we will show that, on average, the number of nodes eliminated in edge selections is 5. For that purpose, we use a potential function, $P(U)$, defined below.

$$P(U) = \lceil |U| - \lfloor \frac{|U_1|}{4} \rfloor - new(U) \rceil \tag{3.1}$$

where

$$new(U) = \begin{cases} 1, & \text{if } U = V. \\ 0, & \text{otherwise.} \end{cases}$$

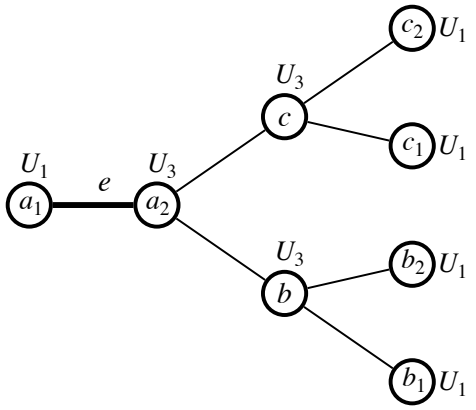

Figure 3.3: Figure to show the elimination of 4 nodes after selecting edge e

and U_1 denotes the subset of U consisting of nodes with exactly one neighbor in U. Note that the initial and final size of U_1 is zero. Therefore, it suffices to use the following selection rule:

Select an edge such that $P(U)$ decreases by at most 5.

In the next section, we will show that whenever $U \neq \emptyset$, there exists an edge that satisfies the selection rule stated above. This implies the following:

Theorem 3.1.2 *There exists a linear time algorithm that finds an induced matching of size at least $\frac{2}{3}$ opt in a 3-regular graph with girth at least 6, where opt is the maximum size of an induced matching.*

Proof. Since the initial value of the potential function $P(U)$ is $|V| - 1$ and the final value of the potential is 0 (as discussed before), with Lemma 2.1.1, our algorithm produces an induced matching of size at least $\frac{2}{3}opt$.

Let $P'(U) = |U| - \lfloor |U_1|/4 \rfloor - new(U)$. Note that $P(U) = \lceil P'(U) \rceil$. For each edge, we will maintain $drop(e)$ which is the change in $P'(U)$ that would result from selecting that edge. Moreover, for each value of $drop(e)$ we will have the doubly linked list of edges with that value. Given that data structure, the edge selection of e with the sufficiently low $drop(e)$ is easy. Furthermore, the number of updates that this data structure requires after a selection can be bounded by a constant. The $drop(e)$ is determined by the number of nodes in e, $N^1(e)$ and $N^2(e)$ that belong to U_1, U_2 and U_3. A selection changes those numbers for at most 125 edges (see Figure 3.4). ❐

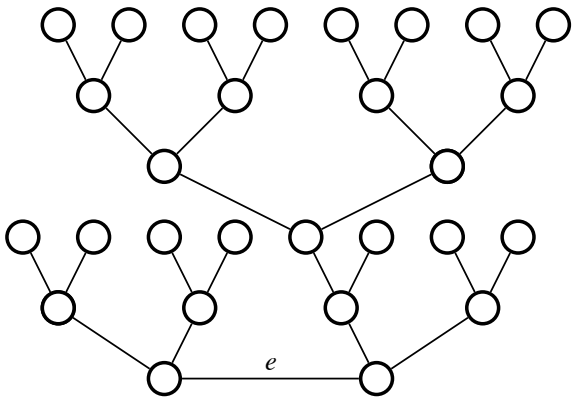

Figure 3.4: This figure shows the selection of e and the resulting elimination modifies $drop(e)$ for at most 125 edges. There are 14 nodes(starting from the top) multiplied by 8 nodes plus 14 - 1 = 125 edges.

3.2 Greedy algorithm part 2 - Rules to make a selection if we are given an input graph

Assumption: There are no cycles of length 5 or less in the input graph, i.e. the girth of input graph, $G(V, E) \geq 6$.

Lemma 3.2.1 *The first edge selection done on the input graph G eliminates exactly 6 nodes.*

Proof. For the first selection, all nodes have 3 edges connected to them (degree of each node is 3 since we start with a 3-regular graph). Keeping in mind the elimination rules in Section 3.1, selecting an edge would eliminate 6 nodes. 2 nodes would be eliminated by rule(a), and 4 nodes are eliminated by rule (b). No nodes are eliminated by rule (c) since there are no nodes of degree 1 in the graph. Figure 3.5 shows the nodes eliminated along with the types of elimination (type (a), type (b) or type (c)).

Figure 3.5 is representative of all possibilities of the first selection. This is because the first selection is done on the input graph which is 3-regular and selecting any edge would produce the same figure and the same analysis would be done as in Figure 3.5. As we have explained the elimination of 6 nodes in the Figure 3.5, lemma 3.2.1 is proven. ❒

Lemma 3.2.2 *If $U \neq \emptyset$, then the induced graph, $G[U]$, contains an edge that can be eliminated with a drop of at most 5 in the potential function $P(U)$.*

Proof. To prove this lemma, we will need to consider a number of cases.
When $U = V$, the drop(e) = 5 for every edge in the graph because we eliminate 6 nodes. This brings the potential down by 6, however $-new(U)$ increases the potential by 1 and U_1 remains empty. Thus, the first selection does not violate the selection rule. Lemma 3.2.1 proves how the first selection eliminates exactly 6 nodes.

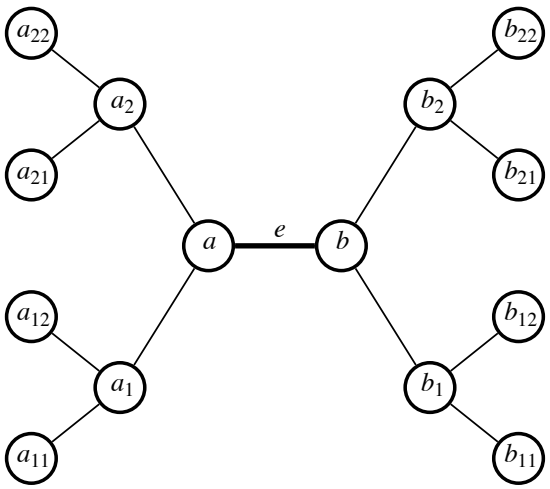

Figure 3.5: Figure to show the elimination of exactly 6 nodes(a, b, a_1, a_2, b_1, b_2) after first edge selection. In this figure, all nodes are in U_3 since the input graph is 3-regular.

When $U \neq V$ and $U_1 = \varnothing$, we choose an edge with a node from U_2. Hence, eliminating at most 5 nodes. It will suffice to show that the resulting size of U_1 will be at most 3, so $-\lfloor \frac{|U_1|}{4} \rfloor$ remains zero.

When $U_1 \neq \varnothing$, we select an edge with a node from U_1 and we will assure that the $drop(e)$ can be at most 5, as either drop in $|U|$ is 6, but then $|U_1|$ drops by 4 or drop in $|U|$ is 5, but then $|U|$ cannot increase. The drop in $|U|$ could also be 4 or less, however the increase in $|U_1|$ can be at most 4. This proof of Lemma 3.2.2 is an overview of the detailed case analysis that follows in the next subsection. ❒

3.2.1 Case Analysis for determining edge to be selected

Note: As mentioned earlier, we assume that the graph has a girth of 6 or greater. In other words, there is no cycle in the graph whose size(number of edges in the cycle) is 5 or smaller.

Definition A neighborhood N of an edge e is defined as the set of nodes that could be possibly eliminated by any rule of elimination. In other words, both nodes of edge e and all nodes that have an edge adjacent to edge e.

Definition The set N_1 is the set of all nodes in U_3 that have one of its neighbor as a node in U_1, where U_3 represents the set of nodes with 3 neighbors (3 other nodes connected to the node) and U_1 represents the set of nodes with 1 neighbor in the current residual graph.

For selecting an edge in a neighborhood N, we first check for nodes with the smallest degree. Suppose we name this node, p. We then select an edge of p with the other node of the smallest degree. In the situation of having multiple edges of the smallest node degrees, we choose the edge whose higher-degree node is connected to the least number of U_3 nodes. This particular criteria of selection can be understood once we look at the different cases. On this note, for selecting an edge we follow the priority list below:

- $U_1 - U_1$ edge
- $U_1 - U_2$ edge
- $U_1 - U_3$ edge
- $U_2 - U_2$ edge
- $U_2 - U_3$ edge
- $U_3 - U_3$ edge

It is noted that if we analyze an edge case, we assume that none of the other edge cases above it in priority are present in the residual graph being considered since we would not be following the priority list.

The first selection is randomly done since selecting any edge would drop $P(U)$ by 5 as described by Lemma 3.2.2. After the first selection, U_2 will be non-empty. The following cases are considered as subsequent edges are selected after the first selection:

1. $U_1 - U_1$ edge: This is a trivial case in which we eliminate 2 nodes and do increase $-|U_1|$ by 0.5. This is the best case possible. This case is shown in Figure 3.6.

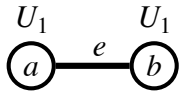

Figure 3.6: Figure to show the $U_1 - U_1$ edge case.

2. $U_1 - U_3$ edge: We will discuss this case before the $U_1 - U_2$ edge case as the $U_1 - U_3$ edge case supports the explanation of $U_1 - U_2$ edge case. We give preference to the edge whose U_3 node has the most number of U_1 neighbors. When we have all three U_1 neighbors attached to the U_3 node, we have a favorable scenario since we drop the potential by exactly 3.25 ($|U|$ drops by 4 and $-|U_1|$ increases by 0.75). When the U_3 node has two U_1 neighbors, in the worst case scenario $|U|$ drops by 6 and $-|U_1|$ increases by 1 which would reduce the potential by 5 as shown in Figure 3.7. When U_3 has only one U_1 neighbor, we have a number of sub cases to consider. When we have 3 N_1 nodes in a chain, the worst possible case is when a U_3 node is attached to the N_1 node being eliminated just as shown in Figure 3.8. We eliminate 5 nodes (namely b,c,c',d,d'). We create 2 U_1 nodes, however we also eliminate two U_1 nodes. The potential function, $P(U)$, decreases by 5. When we have a chain of 2 N_1 nodes, there are 2 worst case scenarios. When we have a $U_2 - U_2$ node pair attached to the $N_1 - N_1$ chain, we eliminate 5 nodes, create 2 U_1 nodes and eliminate 2 U_1 nodes. Thus, we increase the potential by 5. The same is true when a U_3 node, attached to two U_2 nodes, is attached to the $N_1 - N_1$ chain. The potential drops by 5. These two situations are shown in Figure 3.8. The last case is when there is an isolated N_1 node. The worst scenario is shown in Figure 3.8. Four nodes are eliminated and we can create at most 4 U_1 nodes. $|U|$ drops by 4 and $-|U_1|$ drops by 1. Thus, the potential drops by 5.

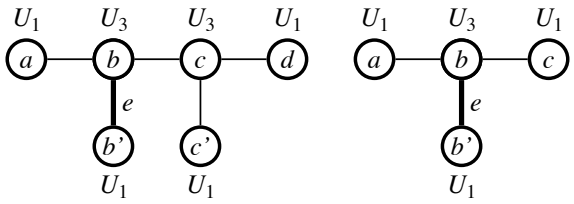

Figure 3.7: Figure to show the case of two U_1 neighbors connected with U_3 in $U_1 - U_3$ case. In the left diagram, the potential drops by 5 and in the right diagram the potential drops by 3.25.

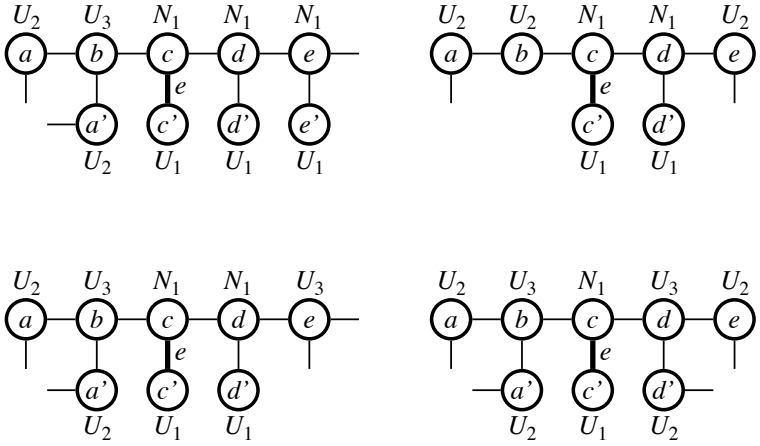

Figure 3.8: Figure shows the different cases of $U_1 - U_3$ edge case with one U_1 neighbor. The selected edge is e. In this figure the nodes that belong to the set $U_3 - N_1$ are marked in U_3.

3. $U_1 - U_2$ edge: This case is simply the $U_1 - U_3$ edge case with 2 U_1 neighbors without one of the neighbors. This is to say that since we have know that the $U_1 - U_3$ edge case with two neighbors does not violate our selection rule, if we just assume that one U_1 neighbor is not present to make all those cases into a $U_1 - U_2$ edge case, we have explained the $U_1 - U_2$ edge case. This is because we will know eliminate one less node which was originally a U_1. So the gain in potential is 0.75. Hence, this case has a greater potential by 0.75 than the $U1 - U_3$ case with two U_1 neighbors. This concept is shown in the Figure 3.9 below. Due to this concept, for every $U_1 - U_2$ edge case we have an increase of 0.75 in potential which explains the last term in the potential function, $P(U)$.

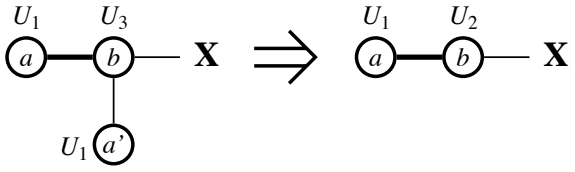

Figure 3.9: Figure to show the transformation of $U_1 - U_3$ edge case with two U_1 neighbors to the $U_1 - U2$ edge case. The 'X' represents the part of the graph remaining from the $U_1 - U_3$ case with two U_1 neighbors.

4. $U_2 - U_2$ edge: When we select a $U_2 - U_2$ edge, we are certain that $U_1 = \emptyset$, otherwise

we would violate the priority list specified. On this note, the maximum number of nodes eliminated on selecting a $U_2 - U_2$ edge is 4. We can create a maximum of 4 U_1's as the worst case, but even if 4 U_1's are created, we do not violate the selection rule ($|U|$ drops by 4 and $-|U_1|$ drops by 1). Thus, the potential function, $P(U)$, would decrease by exactly 5.

5. $U_2 - U_3$ edge: When we select a $U_2 - U_3$ edge, we select the edge that has the U_3 node with the most number of U_2 neighbors. It is noted that $U_2 - U_2$ edges are not allowed in the neighborhood as we would select the $U_2 - U_2$ edge rather than the $U_2 - U_3$ edge. With this observation, we see that the number of nodes eliminated when the U_3 node has 3 U_2 neighbors or 2 U_2 neighbors or 1 U_2 neighbor is 5. The number of U_1's created are at most 2,3 or 2 depending on if the U_3 node has 3,2 or 1 U_2 neighbor respectively. We note that this edge case is only chosen if no other edge cases are available that are of greater priority than this edge case. This explains the small number of possible edge cases. Figure 3.10 shows the different scenarios.

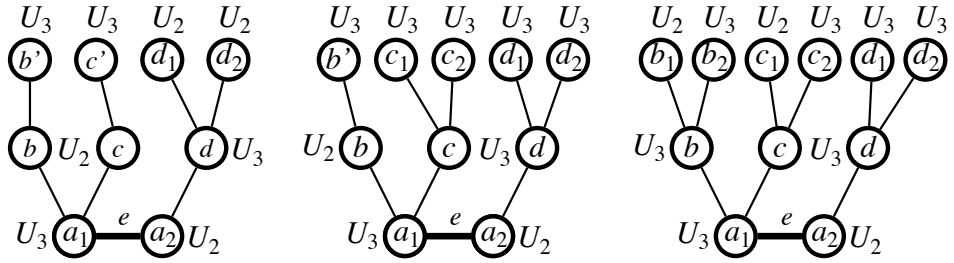

Figure 3.10: Figure shows the different cases of 3,2 or 1 U_2 neighbors for the $U_3 - U_2$ case. We eliminate the nodes of edge e (type (a) elimination). We also eliminate nodes connected to the nodes of edge e (type (b) elimination). Hence, the degree of "upper row" of nodes in the graphs are all reduced by 1.

6. $U_3 - U_3$ edge: This case happens in the beginning as the first selection and has been considered in Lemma 3.2.2.

This concludes the explanation of all the cases required in the analysis of proving the approximation ratio to be at most $\frac{3}{2}$ for 3-regular graphs.

3.2.2 Pseudo code for the greedy algorithm

With the completion of the case analysis, we have proved Lemma 3.2.2 in detail. We can now document the pseudo code for the greedy algorithm that would be used to find the solution of the maximum induced matching problem for 3-regular graphs of girth greater than 6 such that the approximation ratio is at most $\frac{3}{2}$.

We declare V to be the input graph, U to be the residual graph and M to be the matching (solution).

Input: $G = (V, E)$.
$U = V$

$M = \varnothing$
while($U \neq \varnothing$)
    Select the edge e in U according to the priority list stated above and the case analysis.
    This edge is added to M.
    Eliminate the nodes and edges from U according to the elimination scheme.
    In other words, a node, x, is eliminated in the residual graph, U, if x falls into one of the types of eliminations (type (a), type (b) or type(c)).
The solution (matching) is stored M.

The program would begin with the declaration of U which would store the residual graph after every selection and update itself until we find the solution. Therefore, U is initialized to be the input graph, V. We also declare M for storing the solution to the Maximum Induced Matching problem for the particular input graph. M is initialized to be the null set as we have not made a selection yet.

After declaring and initializing the needed variables, we go into a while loop that ensures that edge selections are made until the residual graph is empty. This would mean that we have eliminated all nodes and we have produced an induced matching. In the while loop, we select the edge that is most appropriate for the current residual graph, U. We store this edge in the solution variable, M,. After this, we eliminate the nodes and edges to update U for the next selection. At the end of the loop, the solution is stored in M.

We cannot present a worked out global example in view of the complexity of producing a 3-regular input graph with a girth of 6 or more. This is the main reason why we select analyze each case by understanding the local area of several different parts of the current residual graph. This also explains why problem becomes complex if we tried to consider the entire graph and all the possibilities for producing the induced matching.

Chapter 4

Conclusion

The work here shows a careful analysis of a greedy algorithm that establishes a very efficient approximation algorithm. A complete case analysis showed that we can maintain the reduction of the potential function by at most 5 which supports the elimination of 5 nodes on average when an edge selection is made. As a result, we have proven that an approximation ratio of $\frac{3}{2}$ can be obtained for 3-regular graphs that have a girth of at least 6.

Clearly, the assumption that the graph is 3-regular is not essential for our algorithm because it runs in the residual graph where any node degree is allowed, provided the degree is at most 3. Future work on this topic would include the extension of the technique to obtain $\frac{d}{2}$ approximation for every family of d-regular graphs by showing the reduction in the potential function by at most $2d - 1$. As explained earlier, to accomplish the reduction of the potential function by at most $2d - 1$, we would consider "DoubleStars" of size $2d - 1$ edges. This would lead to the extension of the analysis to obtain a result in a more general setting of "degree-d graphs" in which d is the maximum degree of any node in the input graph G.

Bibliography

[1] M.R. Garey and D.S. Johnson. Computers and Intractability: A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York, 1979.

[2] R.E. Tarjan, A.E. Trojanowski, Finding A Maximum Independent Set, report STAN-CS-76-550, June 1976, CS Department, Stanford University.

[3] Tang Jian, An O($2^{0.304n}$) Algorithm for Solving Maximum Independent Set Problem, in IEEE Transactions on Computers, vol. C-35, no. 9, pp. 847-851, Sept. 1986.

[4] J. Håstad. Clique is Hard to Approximate within $n^{1-\epsilon}$, Acta Mathematica, 182:105-142, 1999.

[5] M. Sviridenko and J. Ward, Large neighborhood local search for the maximum set packing problem. ICALP 2013, 792-803.

[6] W.Duckworth, D. Manlove and M.Zito, On the approximability of the maximum induced matching problem J. Discrete Algorithms 3(1):79-91, 2005.

[7] Zhenyao Li, On Greedy Algorithm for approximating Maximum Induced Matching, Master's Thesis, Department of Computer Science and Engineering at The Pennsylvania State University, 2015.

Chaitanya Patel

Current Address
152 Atherton Hall,
University Park, PA 16802

(717) 808 – 8096 | cop5228@psu.edu

Permanent Address
15 Eastbrook Road,
Ronks, PA 17572

EDUCATION

The Pennsylvania State University, University Park, PA — June 2012 - May 2016
Schreyer Honors College
Bachelor of Science in Mathematics
Bachelor of Science in Computer Engineering
Minor in Statistics

RELEVANT COURSES

- Linear Algebra
- Data Structures and Algorithms
- Stochastic Modeling
- Probability and Stochastic Processes
- Numerical Analysis
- Linear Programming and Game Theory

EXPERIENCE

Honors Thesis and Research — Dec 2014 – April 2016
Graph Theory under Dr. Piotr Berman

- Researching towards designing a greedy algorithm for the maximum induced matching problem
- Crafting a set of priorities to approximate edge selections within the proven upper and lower bounds
- Advancing towards designing a generalized solution for k-regular graphs

IBM Corporation, Poughkeepsie, NY — June 2015 – Aug 2015
Software Developer Intern

- Worked on automating test cases on the robot framework platform of the IBM z Systems hardware management console (HMC)
- Created low-level functions in C++ and Python that support automation via API
- Supported xpaths in XML while delivering tests for different features of the central processing complex (CPC) using the user interface

Shradhdha Hospitality Associates, Ronks, PA — June 2014 – Aug 2014
Assistant Marketing Analyst

- Responsible for the efficient use of sales-related application software
- Maintained statistical and financial records of day-to-day activities using Microsoft Excel
- Analyzed sales figures and assisted in planning allocation of a monthly revenue of $50000

AWARDS AND ACTIVITIES

- Contributed in designing an interface that served as a communication bridge between a door controller and a smart lock for my capstone design project from August 2015 to December 2015
- Vice President of the Penn State Badminton Club from August 2015
- Elected to the Eta Kappa Nu (HKN) computer engineering honor society in March 2014
- Recipient of the Friends of Penn State Scholarship (valued at a total of $12000 across six semesters) and the Lisa Ann Baker Memorial Award (valued at $920) for outstanding academic performance
- Member of Ohana, a fundraising organization, for helping children fight pediatric cancer

COMPUTER SKILLS

Languages : C, C++, Python, Java, MATLAB, R, SAS
Operating Systems : Windows NT, Mac OS X, Linux (Ubuntu/Xubuntu)
Software Packages : Microsoft Office, SimpleScalar, NI Multisim, Logicworks