2012

Extracting large quasi-bicliques using a skeleton-based heuristic

Nick Pappas
Iowa State University

Extracting large quasi-bicliques using a skeleton-based heuristic

by

Nick Pappas

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Oliver Eulenstein, Major Professor
David Fernandez-Baca
Ryan Martin
Giora Slutzki
Sabine Baumann

Iowa State University

Ames, Iowa

2012

Table of Contents:

Abstract

One important computational problem is that of mining quasi bicliques from bipartite graphs. It is important because it has an almost endless number of applications and, in most real world problems, is more appropriate than the mining of bicliques. In my thesis I examine the following: the motivation for quasi bicliques, the existing literature for quasi bicliques, my implementation of a web application that allows the user to compute exact quasi biclique solutions using the biclique formulation and the exact solution algorithm provided by Chang *et al.*[1], and finally a polynomial heuristic algorithm for finding large quasi bicliques in the special case where we have all the biclique subgraphs of a bipartite graph available.

Chapter 1 – Introduction

1.1 Motivation

There is a plethora of real-world problems that can be handled by modeling the problem as a bipartite or k-partite graph problem and then mining one or more biclique subgraphs (or k-cliques in the case of the k-partite graph) from that graph. A biclique is a bipartite graph in which each vertex in one partition is connected to every other vertex in the other partition. Applications amenable to such treatment are mostly applications in business, web mining and bioinformatics as they most naturally can be modeled into a bipartite or k-partite model problem. For example, online retailers like Amazon or New Egg would like to suggest to or present their visitors with products that are most likely to lead to a purchase by using the history of the behavior of the specific visitor in comparison to the history of the behavior of all other visitors (bipartite graph case with the partitioning “visitors”-“products”). Another example would be a web search engine like Google or Bing that relies on advertisements for profit. Such engines would like to show to their users appropriate advertisements based on the user’s history of web searches and visitation of the results as to maximize the advertisement clicks (this is a tripartite case with the partitioning of “users”-“webpages visited after a search”-“advertisements”).

However, perhaps the most natural setting of all for the biclique extraction method is in biological problems and particularly in the problem of predicting protein-protein interaction. Proteins are key factors for life, responsible for almost the totality of biological functions. Proteins are needed for genome expression, for the conversion of the genome from DNA to RNA and from RNA to proteins, for signal transduction, for cell-cell communication, for immunoreactions, etc. Protein-protein interaction prediction can lead to novel drugs, cure of diseases, and more. Thus, determining the

binding site location of proteins (the places in proteomic sequences where proteins actually interact) is one of the most important goals in protein research. The appeal of computational approaches when researching the protein binding sites is that the non-computational approaches (X-ray crystallography, electron microscopy, mass spectrometry, etc.) are both very expensive and very slow and furthermore the number of known proteins is huge.

Not surprisingly, the literature on mining of biclique subgraphs technique is diverse and plentiful: the idea has been used for the co-clustering of groups of words and groups of documents [2], the discovery of web communities by co-clustering groups of users and groups of webpages [3-6], the co-clustering of groups of species and groups of genes for the construction of the tree of life [7-9] and most frequently for the discovery of protein-protein interactions [10-15] or protein function prediction [16, 17], just to mention a few examples. All of the above examples have three common characteristics, discussed below.

First, the graphs that model the problem can be very large. Second, what is actually needed is the enumeration of all *maximal* biclique subgraphs (a biclique subgraph of a bipartite graph G is maximal if and only if it is not a proper subset of any other biclique subgraph of G). Although the problem of finding a maximum vertex biclique given a graph G has polynomial solution [18], the interesting problem in which we are trying to maximize the edges (i.e. the relations) of a biclique given a graph G has been shown to be NP-hard [19]. Third, since all the data are real world data, there are plenty of missing or erroneous values. In the real world we have examined only so many "documents-words" associations, tracked only so many "users-web pages" relationships, we are aware of only so many "shareholders-companies" affiliations, we have tested only so many protein to protein interactions, and so on. Nevertheless, in many of the above applications the requirement of mining a biclique (an "all-to-all" relation) is not needed since we would simply get satisfactory results by examining only

"sufficiently many-to-sufficiently many" relations. This third characteristic of missing data, having erroneous data and not always needing "all-to-all" relations gave rise to the idea of mining quasi-bicliques instead of bicliques.

A quasi biclique (or qbc) is almost a biclique in the sense that we are allowing it to miss a certain number of edges that would be present if it were to be an actual biclique. Although quasi-biclique extraction turns out to also be intractable (for all the various quasi-biclique formulations and definitions), quasi bicliques are of interest because they are more realistic models for handling the real world missing or erroneous data, are more appropriate for the applications that require "sufficiently many-to-sufficiently many" relations and in certain cases can provide better results anyway [20, 21]. However, one of their most obvious difficulties is defining quasi-ness, as the idea of "almost-a-biclique" can be interpreted in many ways. In the next section I will provide some preliminaries, notation and terminology that will be used throughout the thesis, and then I will examine the attempts to define quasi-ness so far in literature.

Finally in the next two chapters, I will describe my own contribution. In chapter 2 I will describe my implementation of a web application that allows the user to compute exact quasi biclique solutions using the biclique formulation and the exact solution algorithm provided by Chang *et al.*[1]. In chapter 3 I will describe a heuristic algorithm that allows the extraction of large quasi bicliques in polynomial time using the concept of a *skeleton* and a feasible circulation characterization of quasi bicliques.

1.2 Preliminaries, notation, terminology and definitions

In this section I will provide the notation, terminology and definitions that will be used throughout the rest of the thesis. Since the definition of quasi-biclique varies depending on the paper I will give the relevant notation for each different definition when needed later in the thesis. Furthermore since particular notation and definitions are needed only for specific sections of the thesis I will simply present them when they are needed.

Unless otherwise noted, all graphs in the thesis are considered to be undirected.

An **undirected graph** denoted as $G(V,E)$ or simply G consists of a non-empty and finite set of vertices V and a set of edges $E = \{\{u,v\}\ s.t.\ \mathrm{u} \neq \mathrm{v} \text{ and } \mathrm{u,v} \in V\}$. Two vertices are **adjacent** if they are the endpoints of an edge, i.e. if there exists an edge $\{u,v\}$ that connects them.

For a vertex v we denote the **neighbor set** of v (the set of all vertices adjacent to v) as $\Gamma(v)$.

For a vertex set $S \subseteq V$ we denote by $\Gamma(S) \triangleq \bigcap_{u \in S} \Gamma(u)$ the set of all vertices each of which neighbors every vertex in S.

A graph $g(V',E')$ is a **subgraph** of G if $V' \subseteq V$ and $E' \subseteq E$.

A subgraph $g(V',E')$ of a graph $G(V,E)$ is said to be **induced** if for any pair of vertices $u,v \in V'$ the edge $\{u,v\} \in E'$ if and only if $\{u,v\} \in E$.

A **bipartite graph** denoted by $G(U \cup V, E)$ is a graph whose vertices can be partitioned in two disjoined sets of vertices U and V and an edge set $E = \{\{u, v\}\ s.t.\ u \in U\ and\ v \in V\}$.

A **complete bipartite graph** or **biclique** is a bipartite graph $G(U \cup V, E)$ in which for any $u \in U$ and $v \in V$ there exists an edge that connects the two, i.e. $\{u, v\} \in E$.

We say that a bipartite graph $G(U \cup V, E)$ **contains** a biclique if we can find subsets $U' \subseteq U$, $V' \subseteq V$ such that the subgraph G' induced by $U' \cup V'$ is complete.

A biclique is called **maximal** if it is not included in any other biclique.

A **weighted biclique** $G(U \cup V, E, w)$ is a biclique $G(U \cup V, E)$ plus a weight function w that assigns a weight on each edge $w: E \to [0,1]$

1.3 Quasi – biclique literature review

In this section I will provide a review of the literature on quasi bicliques. I will attempt to make clear what are the differences as well as the similarities between the various definitions and approaches to the problem.

1.3.1 γ - quasi bicliques

One first attempt to define quasi-ness is by considering the density of a graph. A graph (or a bipartite graph) is a quasi clique (or quasi biclique) if it is "dense enough", i.e. if it is not missing too many of the edges that would otherwise make it a clique (or biclique). However the density based approach carries with it the same flaws as quasi-ness. It seems that defining the "dense-enough" is a subjective matter. Never the less Abello *et al*. [22] provide such an approach.

Notation:

Let $G(V,E)$ be a graph and $S \subseteq V$. By G_S we denote the subgraph of G induced by S .

Definition 1.3.1.1: $\gamma - density$

A graph $G(V,E)$ is $\gamma - \mathbf{dense}$ if $|E| \geq \gamma \binom{|V|}{2}$.

Definition 1.3.1.2: $quasi - clique$

A $\gamma - \mathbf{clique}$ or **quasi-clique** is a set of vertices $S \subseteq V$ such that the induced graph G_S is both connected and $\gamma - \text{dense}$.

Problem 1.3.1.1: The maximum quasi-clique problem

Input: A graph $G(V,E)$ and $0<\gamma\leq 1$.

Query: Find a quasi-clique of maximum cardinality.

Abello *et al*. observe that when $\gamma=1$ the problem is NP-hard since the question then is to find the maximum clique in a graph; a known NP-hard problem. Furthermore they notice that there can be no polynomial approximation algorithm that can approximate the maximum clique size within a factor of n^{ε} $(\varepsilon>0)$ unless $P=NP$ [23-26]. They proceed by introducing the notion of the potential of a set of vertices R in respect to a disjoint set S where $R,S\subseteq V$ for a graph $G(V,E)$.

Definition 1.3.1.3: *potential*

Let $G(V,E)$ be a graph and $R\subseteq V$ be a quasi-clique. We define the potential of a set R to be:

$$\varphi(R)=|E(R)|-\gamma\binom{|R|}{2}$$

Definition 1.3.1.4: *potential of a set R with respect to a disjoint set S*

Let $G(V,E)$ be a graph and S,R with $R,S\subseteq V$ be two disjoint quasi-cliques. We define the potential of a set R with respect to S to be:

$$\varphi_S(R)=\varphi(R\cup S)$$

Problem 1.3.1.2:

Input: A graph $G(V,E)$, $0<\gamma\leq 1$ and a quasi-clique S.

Query: Find quasi-clique R that is as large as possible such that $S\cup R$ is also a quasi-clique.

In other words we are looking for sets R with large potential (i.e dense enough). In the ideal case the cardinality of R would be maximum. However as we discussed earlier the problem of finding a quasi-clique of maximum cardinality (problem 3.1.1) is NP-hard. Thus they take the approach of constructing incrementally a maximal quasi-clique by using a heuristic based on a greedy randomized adaptive search procedure algorithm (GRASP) [27].

So far we have seen nothing regarding bicliques and quasi bicliques; however everything discussed so far applies to bipartite graphs as well. Given a bipartite graph $G(U \cup V, E)$ their heuristic finds a balanced quasi biclique in $O(|M\alpha| \cdot |U \cup V|^2)$ where $|M\alpha|$ is the size of a maximum matching in G.

1.3.2 (non symmetrical) ε - quasi bicliques

Motivated by the Conjunctive Clustering problem, Mishra *et al*.[28] provide a different definition of a quasi biclique, the ε-clique. A conjunctive cluster is a conjunction of attributes c together with the points Y in the data set that satisfy the conjunction c. In the Conjunctive Clustering problem the task is the identification of as long as possible conjunctive clusters that cover an as dense as possible region of space. The problem translates naturally to a biclique in a bipartite graph formulation.

Let $G(U \cup V, E)$ be a bipartite graph where U is the number of points in the dataset to be clustered and V is the set of attributes or dimensions. For any two vertices $u \in U$ and $v \in V$ we have an edge $\{u, v\}$ if the v^{th} dimension is 1 (i.e. if u has the attribute v). Given this formulation, it follows that the best conjunctive clustering corresponds to a maximum edge biclique. However as we already discussed, finding the maximum edge biclique is both NP-hard and difficult to approximate and therefore Mishra *et al*. relax the problem by requiring that in any given subgraph $G'(U' \cup V', E')$

returned by an algorithm we ask that each vertex $v \in V'$ is connected to most vertices $u \in U'$.

They define the ε-clique as follows:

Definition 1.3.2.1: *ε-biclique*

Let $G(U \cup V, E)$ be a bipartite graph and let $U' \subseteq U$ and $V' \subseteq V$. Given $0 < \varepsilon \leq 1$ we say that the vertex set $U' \cup V'$ is **ε-close** to being a biclique if every vertex in U' is adjacent to at least (1-ε) of $|V'|$. The induced subgraph $G'(U' \cup V', E')$ in that case is called an **ε-biclique**.

Recall that by $\Gamma(S) \triangleq \bigcap_{u \in S} \Gamma(u)$ we denote the set of all vertices each of which neighbors every vertex in S .

Definition 1.3.2.2

Let $G(U \cup V, E)$ be a bipartite graph, let $S \subseteq U \cup V$ and let $0 < \varepsilon \leq 1/2$.

We let $\Gamma_\varepsilon(S) \triangleq \{w : |\Gamma(w) \cap S| \geq (1-\varepsilon)|S|\}$ denote the set of vertices that neighbor all but an **ε-fraction** of S .

Problem 1.3.2.1

Input: A bipartite graph $G(U \cup V, E)$, $0 < \varepsilon \leq 1/2$ and a small constant $b > 0$.

Query: Find a subset V' such that the **ε-biclique** induced by $(\Gamma_\varepsilon(V'), V')$ is at least $(1 - b\varepsilon)$ times as large as the maximum biclique for a small constant b .

Note that in the case that where $b = 0$ the ε-biclique will contain the same number of edges as a biclique.

The algorithm provided in this paper does not output the ε-biclique subgraph $G'(U' \cup V', E')$ but rather only the set of vertices V' from which the set of vertices U' is determined implicitly from V', as the elements of U' is precisely those elements of U that neighbor at least $(1-\varepsilon)$ vertices in V'. The running time varies greatly depending on the size of U' and V' and can range from linear to exponential. However the biggest flaw of this approach is that the resulting quasi bicliques are not symmetrical (the authors justify the asymmetry by reminding us the ε-cliques serve needs in the context of clustering), a detail that I will talk in more depth in section 3.4.

1.3.3 α - quasi bicliques

The idea behind the α – quasi bicliques, as presented by Yan *et al.* [9] is to use a percentage measurement to define quasi-ness.

Definition 1.3.3.1: *α - extension*

Let $G(U \cup V, E)$ be a bipartite graph and $G'(U' \cup V', E')$ be a biclique subgraph of G. We define an **α-extension** of G' to be the bipartite subgraph of G induced by the vertex sets U_E, V_E $U_E \subseteq U - U'$ and $V_E \subseteq V - V'$ such that at least $a\%$ of the nodes in each of U' and V' are connected to all the nodes in U_E and V_E respectively.

Definition 1.3.3.2: *α - quasi biclique*

Let $G(U \cup V, E)$ be a bipartite graph, $U', U_E \subseteq U$, $V', V_E \subseteq V$ where the subgraph G' induced by (U', V') is a biclique, a be a % percentage and the subgraph induced by (U_E, V_E) be an α-extension of G'. An **α- quasi biclique** is defined as the subgraph of G, G*induced by the ordered pair $(U' \cup U_E, V' \cup V_E)$.

In other words, each vertex in the extension of a biclique will be adjacent to at least α% vertices in the biclique. The idea is to somehow extract all bicliques from a

bipartite graph and then expand each one of them in order to obtain the corresponding maximal α - quasi biclique. There are two main issues with this approach. First, we need to extract all the bicliques of the bipartite graph before we start extending them. This is an issue since we do not have an efficient algorithm to do this. Second, it could be the case that the original graph contains no bicliques but could have useful quasi-bicliques nevertheless. However since both of these issues are issues that my heuristic has in common with this approach I will address them in detail in chapter 5.

1.3.4 error tolerant ε – quasi bicliques

Another idea is to approach quasi-ness from an error tolerance perspective. Sim *et al*. [29] define their quasi biclique version based not on how many edges we would like to have but rather on how many edges we are allowed to be missing.

Definition 1.3.4.1 : *error tolerant ε – quasi biclique*
Let $G(U \cup V, E)$ be a bipartite graph and $\varepsilon \in \mathbb{Z}^+$. We say that G is an **error tolerant ε – quasi biclique** if for every $u \in U$, $|V| - |\Gamma_V(u)| \leq \varepsilon$.

Again what we need is given a bipartite graph $G(U \cup V, E)$ to find all the maximal error tolerant ε – quasi bicliques. The parameter ε is called the **error tolerant threshold**.

Sim *et al*. are introducing two quite useful notions on the error tolerance of QBCs in order to characterize them, and they proceed in a comparison of their approach to the other approaches in the literature that are aware based on how they perform based on these notions.

Definition 1.3.4.2 : *symmetrical*
We say that the error tolerance in a qbc is symmetrical if vertices in both sides of the qbc can tolerate missing edges.

Based on the above definition, it is clear that the approach described earlier in 3.2 is not symmetrical since there is no imposing of any requirement as to how many edges can be missing on one of the two sides. The approaches described in 3.1 and 3.3 are both symmetrical however.

Definition 1.3.4.3 : *balanced*

We say that the error tolerance in a qbc is balanced if every vertex in the quasi biclique can tolerate up to the same threshold of missing edges.

Based on 1.3.4.3 the approaches described in all the previous sections are clearly not balanced. The authors provide a useful table comparing the approaches reproduced slightly altered in table 1.

Table 1. Comparison of different types of quasi-bicliques and their algorithmic approach.

Definition	Type	Symmetrical	Balanced	Algorithm
Error tolerant - qbc	Maximal	Yes	Yes	Complete
γ-qbc	Density	Yes	No	Greedy
ε-qbc	Non-maximal	No	No	Greedy
α-qbc	Maximal	Yes	No	Complete

What follows is the description of an algorithm that they call MQBminer (Maximal Quasi Biclique miner) with an unreported running time that is nevertheless depending on the minimum size threshold (how small can we allow the sets of vertices to be), the error tolerant threshold and the density of the graph. The mining of small

maximal qbcs that allow large ε is deemed to be very expensive. Similarly the computation becomes very expensive for graphs that are dense and we impose a small minimum size threshold.

1.3.5 δ - quasi bicliques

Another idea of quasi-ness is to use the degree of the participating nodes in each partition of the quasi bicliques as in Liu *et al.*[21]

Definition 1.3.5.1: *vertex degree*

Let $G(U \cup V, E)$ be a bipartite graph. For a vertex $u \in U$ and a vertex set $V' \in V$, the **degree of** u **in** V' is denoted by $d(u,V') = |\{v \text{ s.t } v \in V' \text{ and } \{u,v\} \in E\}|$.

Definition 1.3.5.2: *δ - quasi biclique*

Let $G(U \cup V, E)$ be a bipartite graph, and let $0 < \delta \leq 1/2$. G is a **δ – quasi biclique** if for each $u \in U$ $d(u,V) \geq (1-\delta)|V|$ and for each $v \in V$ $d(v,U) \geq (1-\delta)|U|$.

Using the above definition, Liu *et al.* formulate the following two problems.

Problem 1.3.5.1: *δ - quasi biclique problem*

Input : A bipartite graph $G(U \cup V, E)$, and $0 < \delta \leq 1/2$.

Query: Find $U' \subseteq U$ and $V' \subseteq V$ s.t. the $U' \cup V'$ induced subgraph of G is a δ - quasi biclique and $|U'| + |V'|$ is maximized.

Problem 1.3.5.2: *Balanced δ - quasi biclique problem*

Input: A bipartite graph $G(U \cup V, E)$, and $0 < \delta \leq 1/2$.

Query: Find $U' \subseteq U$ and $V' \subseteq V$ s.t. the $U' \cup V'$ induced subgraph of G is a δ - quasi biclique and $|U'| = |V'|$ is maximized.

The authors proceed in showing that both problems are NP-Hard using a reduction from the Exact Cover by 3-Sets problem which was shown to be NP-Hard by Karp[30] in the first case and from the perfect 3 cover problem which was also shown to be NP-hard in the second case[31].

Finally they describe a greedy heuristic implemented in Java that runs in $O(n^3)$.

1.3.5.1 Approximating δ - quasi bicliques

Wang in [32], describes a polynomial time approximation scheme to obtain a quasi biclique U', V' for $U' \subseteq U$ and $V' \subseteq V$ with $|U'| \geq (1-\varepsilon)|U_{opt}|$ and $|V'| \geq (1-\varepsilon)|V_{opt}|$ such that any vertex $u \in U'$ is adjacent to at least $(1-\delta-\varepsilon)|V'|$ vertices in V' and any vertex $v \in V'$ is adjacent to at least $(1-\delta-\varepsilon)|U'|$ vertices in U'. However the paper is not very well written and the author provides no proofs.

1.3.6 αβ – weighted quasi bicliques

The αβ – weighted quasi biclique approach is quite different from the rest in that it both allows an intended possible asymmetry between the two partitions, and also is the only approach that considers the edge weights in a bipartite graph. Chang *et al.*[1] define the αβ – weighted quasi biclique as follows:

Definition 1.3.6.1: *αβ – weighted quasi biclique*

Let $G(U \cup V, E, w)$ be a weighted bipartite graph, and let $a, \beta \in [0,1]$. An **α,β-weighted quasi biclique** in G is the subgraph of G induced by the pair (U', V') with $U' \subseteq U$ and $V' \subseteq V$ that satisfies the following properties:

1. $U', V' \notin \{\varnothing\}$

2. $\forall u \in U' : \sum_{v \in V'} w(u,v) \geq a \times |V'|$
3. $\forall v \in V' : \sum_{u \in U'} w(u,v) \geq \beta \times |U'|$

Definitions 1.3.6.2: *aβ – weighted quasi biclique weight*

The **weight** of an α,β-weighted quasi biclique is defined as the sum of the weights of all its edges. An α,β-weighted quasi biclique of a weighted bipartite graph $G(U \cup V, E, w)$ is **maximum weighted** if its weight is at least as much as the weight of any other α,β-weighted quasi biclique in G .

Based on the above definitions Chang *et al.* formulate the following problems:

Problem 1.3.6.1: *Maximum weighted aβ – weighted quasi biclique*

Input: A weighted bipartite graph $G(U \cup V, E, w)$ and values $a, \beta \in [0,1]$.

Query: Find a maximum weighted aβ – weighted quasi biclique.

Problem 1.3.6.2: *Inclusion*

Input: A weighted bipartite graph $G(U \cup V, E, w)$, values $a, \beta \in [0,1]$ and a pair of vertex sets $P \subseteq U$ and $Q \subseteq V$.

Query: Find an aβ – weighted quasi biclique that includes P, Q .

Problem 1.3.6.3: *Included Maximum Weighted Quasi Biclique*

Input: A weighted bipartite graph $G(U \cup V, E, w)$, values $a, \beta \in [0,1]$ and a pair of vertex sets $P \subseteq U$ and $Q \subseteq V$.

Query: Find a maximum weighted aβ – weighted quasi biclique that includes P, Q .

All three problems are proven to be NP-Complete. The first by a reduction from the maximum edge biclique problem. The second by a rather convoluted series of

reductions: A special case of the inclusion is considered (the one sided existence problem), then a modified version of the weighted quasi biclique problem is used (the modified one sided existence problem). Finally a well know NP-Complete problem, the partition problem, is reduced to the modified one sided existence problem which in turn is reduced to the one sided existence problem which in turn reduces to the inclusion problem.

The authors provide an ILP formulation for solving the maximum weighted quasi-biclique problem inspired by ILP solutions to the knapsack problems as well as an implementation in Python that is using the commercial solver Gurobi and is hosted on a linux machine. However there was no interface provided for using the implementation.

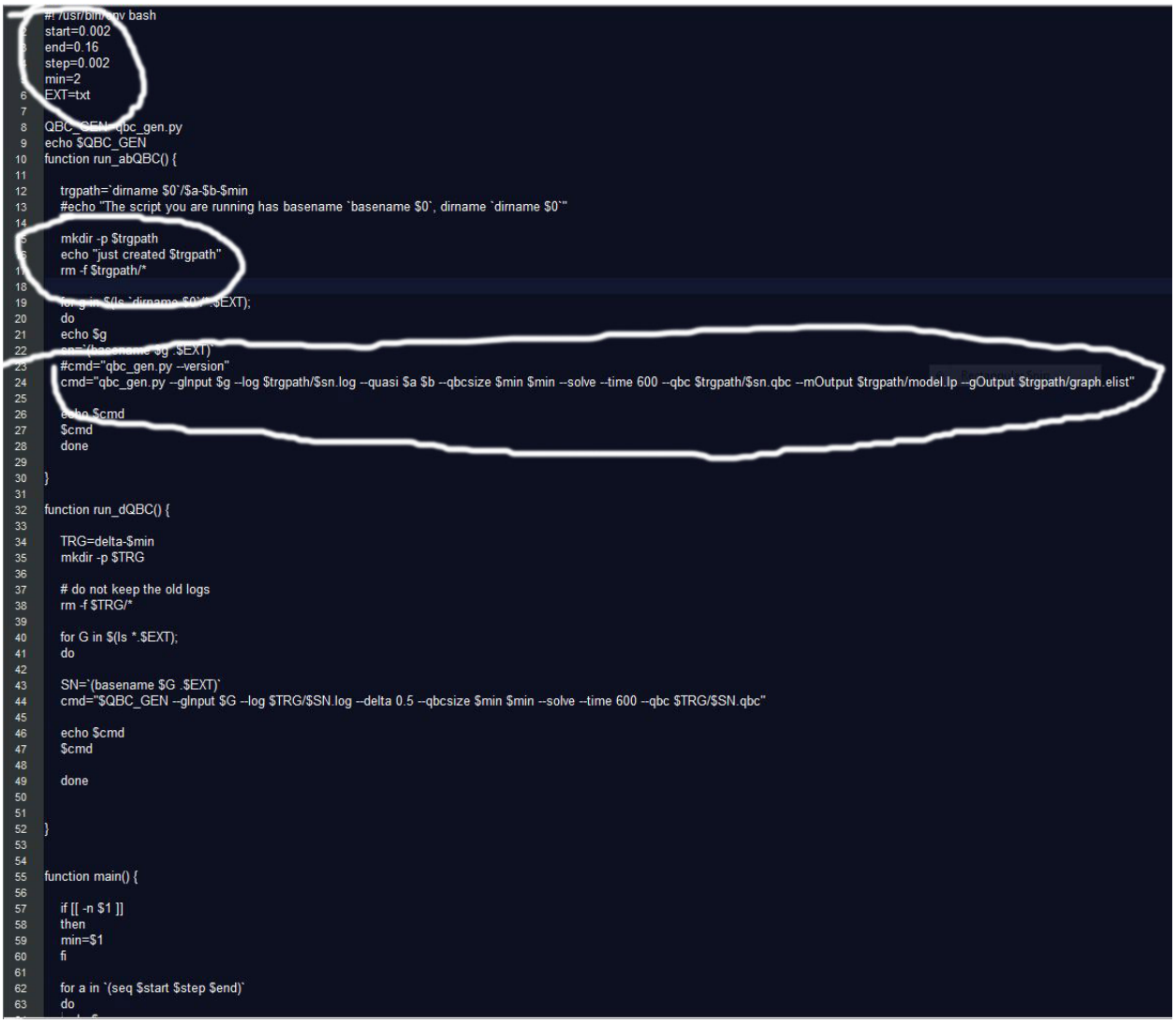
Chapter 2 - αβ – weighted quasi bicliques web application

2.1 The ILP implementation status

The status of the implementation in Chang *et al.* [1] was functional on the one hand but hardly user friendly on the other. In order for the user to use the ILP formulation and the solver, the user should have SSH access to the linux machine hosting the Gurobi solver. Then he or she would have to alter the bash shell script that would run the python program and that would feed the output to the solver every time the change of a parameter was necessary (Figure 2.1.1).

The user would also have to make sure that the input files are in the correct format (e.g. tab separated files vs. comma separated files vs. space separated files, and so on.) The output of the program was simply a bunch of cryptic text files that the user would have to inspect in a text editor, in which he or she would simply look at lines containing xs, ys and numbers (Figure 2.1.2).

The user could not save the work that was already done (so as not to repeat it in the future) by using a meaningful name for it, and finally there was no separation or data privacy between the users as the implementation was meant to be for a single user only. This last observation was very important because it meant that every time a user (e.g. a collaborating professor) would like to analyze some data in order to evaluate the resulting qbcs, or the chosen αβ values, he or she would have to contact us so that we could do the work and then send back to him or her the results (i.e. send back a bunch of text files like the one seen in Figure 2.1.2).



```
#! /usr/bin/env bash
start=0.002
end=0.16
step=0.002
min=2
EXT=txt

QBC_GEN=qbc_gen.py
echo $QBC_GEN
function run_abQBC() {

    trgpath=`dirname $0`/$a-$b-$min
    #echo "The script you are running has basename `basename $0`, dirname `dirname $0`"

    mkdir -p $trgpath
    echo "just created $trgpath"
    rm -f $trgpath/*

    for g in $(ls `dirname $0`/*.$EXT);
    do
    echo $g
    g=`basename $g .$EXT`
    #cmd="qbc_gen.py --version"
    cmd="qbc_gen.py --glnput $g --log $trgpath/$sn.log --quasi $a $b --qbcsize $min $min --solve --time 600 --qbc $trgpath/$sn.qbc --mOutput $trgpath/model.lp --gOutput $trgpath/graph.elist"

    echo $cmd
    $cmd
    done

}

function run_dQBC() {

    TRG=delta-$min
    mkdir -p $TRG

    # do not keep the old logs
    rm -f $TRG/*

    for G in $(ls *.$EXT);
    do

    SN=`basename $G .$EXT)`
    cmd="$QBC_GEN --glnput $G --log $TRG/$SN.log --delta 0.5 --qbcsize $min $min --solve --time 600 --qbc $TRG/$SN.qbc"

    echo $cmd
    $cmd

    done


}


function main() {

    if [[ -n $1 ]]
    then
    min=$1
    fi

    for a in `seq $start $step $end)`
    do
```

Figure 2.1.1. Bash shell script. Circled are the areas that the user would have to alter in order to get different results (e.g. different αβ values, different number of resulting bicliques, different number of minimum vertices per partition, etc.).

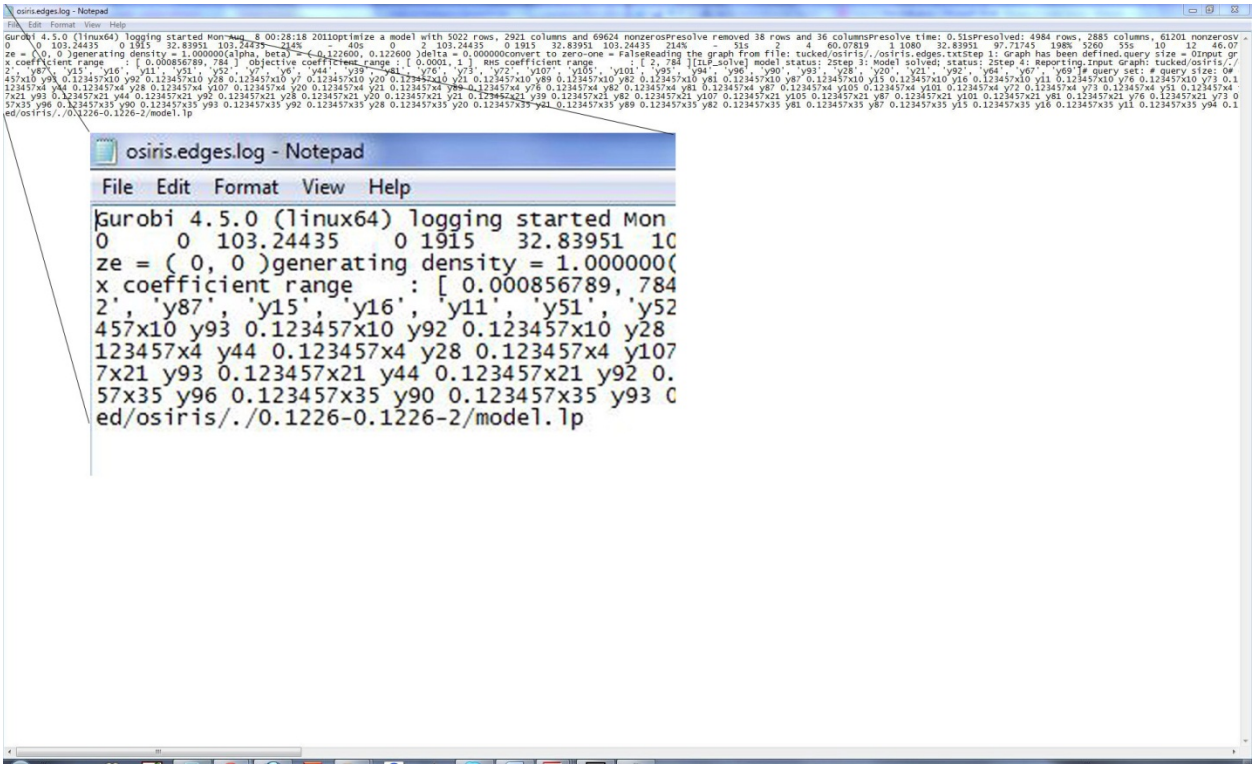


Figure 2.1.2. Gurobi output with detail as seen on a Windows machine

2.2 The application

I designed a web application based on the MVC (Model View Controller) paradigm that has the following characteristics:

a) Multiple users with different user names and passwords are able to use the ILP implementation on their own.

b) The access of the ILP formulation can be done from anywhere through a web page.

c) The results are saved and the users can access them later.

d) The change of the input parameters is as simple as filling a web form.

e) The results can be seen in an interactive graphical interface that will allow the user to get an immediate idea of the quality of the results.

What follows is a short description of the tools used for the application as well as a short user manual.

2.2.1 Tools

For the application I used Java for programming the model and controller part, Flex and JSP for the view part and MySQL as the database of choice for storing the user results. All that the user needs to run the application is a web browser that can show Adobe Flash content (for the FLEX interactive part of the result presentation). The server however requires the installation of Java (for the Java parts), Python (for the Python parts written by Chang *et al.*), Apache Tomcat, as well as Hibernate (for the JSP/servlet parts) , mySQL for the database parts of the problem and of course the Gurobi solver for solving the ILP. The machine available (hiv.cs.iastate.edu) did not have support for Tomcat or mySQL and thus the application could not be deployed on it at the time of this writing. The application thus was not tested in the hiv environment but only on personal machines. Unfortunately, my personal machines do not have access to the Gurobi solver meaning that the solver part of the application had to be replaced with a dummy solver. Nevertheless, the application can be deployed in any server that has all the prerequisites at any time with only minimal effort.

For the Java part of the application, I used a modified version (by myself) of the 0.8.3 iteration of the jgrapht library for graph modeling (the jgrapht library can be found for free at www.jgrapht.org). For the Flex part of the application, I used the Kalileo library provided by Kapit (The Kalileo library can be found for free at www.kapit.fr). The

application is meant to be modular and thus any part of it could be replaced by a different implementation (e.g. a different solver, a non FLEX presentation, etc.). The entire written code was deposited on sourceforge (www.sourceforge.org) and can be accessed by any SVN client (for example directly through Eclipse).

2.2.2 User Guide

The application usage is very simple. The user initially will be asked to provide a user name and password or to register for one (at the point of this writing this was meant to be done freely without the need for a user database administrator). Upon successful login the user has the option to either copy paste an input graph in an input box, or to upload a text file that contains the input graph. In both cases, the correctness of the input is checked and if the input is found incorrect there are two options (the choice was left open to be decided later): either attempt to correct the input or ask the user to retry submitting. The user also provides a name for the current job, value ranges for α and β (an initial value and a stop value), an increment step value for α and β and a minimum threshold value that is required for the returned quasi bicliques. For example an input graph G with α=β=0.01, step value 0.005, stop value of 0.025 and minimum threshold of 3 would return the quasi bicliques extracted from G that have at least 3 nodes in each of their partition and are obtained by using the αβ values: α=β=0.01, α=β=0.015, α=β=0.2 and α=β=0.025.

The resulting quasi bicliques are stored in a database so that the user can avoid redundant calculation and are given to the user as links. Clicking a resulting link takes the user to the interactive graphical representation of that particular result. For example clicking the 0.02-0.02-3 result will take them to the qbc calculated with α=β=0.2 and that contains at least 3 vertices in each partition. The vertices of the resulting quasi biclique are colored red and green for each partition and the weight of both vertices and

edges are color and size coded. Heavier weight nodes (nodes that contributed more in the resulting quasi biclique) are darker and heavier weight edges are bolder. The user can move around each node individually or an entire partition or even the entire graph, he or she can collapse the partitions into a single node for easier handling of very large graphs, and can also zoom in and out. The user is getting some overall info and statistics about the resulting biclique (e.g. parameters used, overall resulting weight, number of edges, etc). Upon setting the cursor on a particular edge or vertex the user gets information about that particular edge or vertex (e.g. weight, vertex degree etc.). Visited vertices (i.e. nodes that were touched by the cursor) are size coded so that the user can see in a glance which nodes are already inspected. Heavier vertices are larger while lighter vertices are smaller. Double clicking anywhere on a node resets the graph to its original form (where the graph fits entirely in the screen and the vertices in the two partitions as well as the two partitions themselves are evenly spaced). The flash interface can be seen in figures 2.2.2.1 through 2.2.2.4.

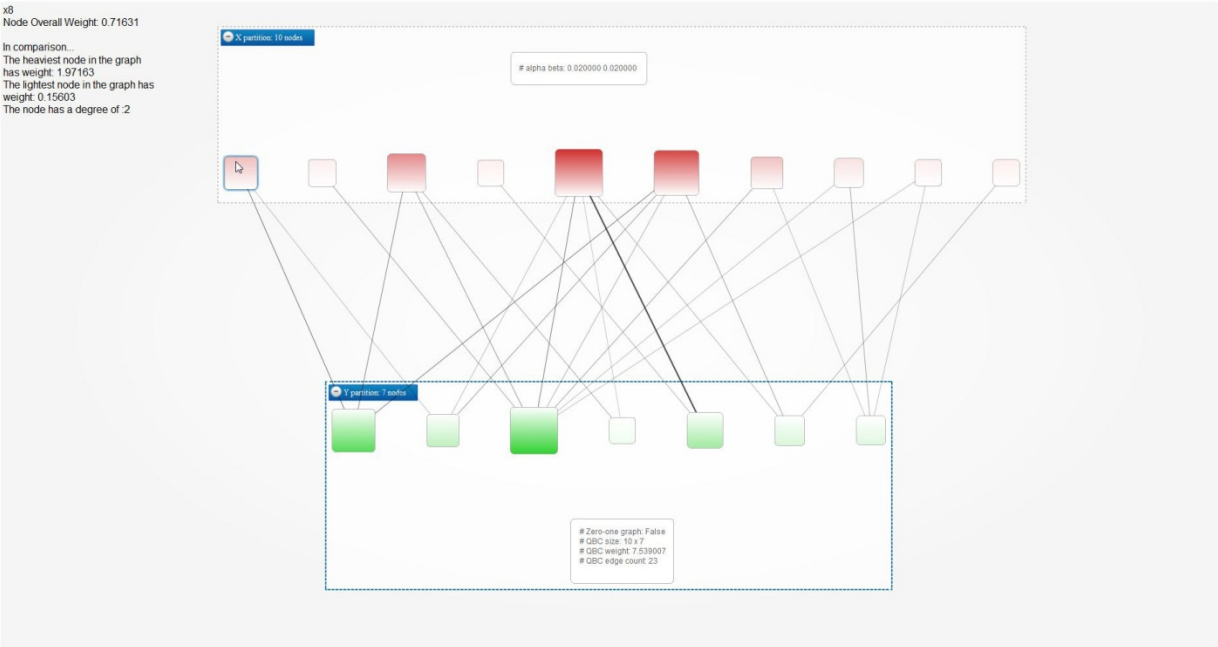


Figure 2.2.1. All the nodes are inspected and thus color and size coded. Information about the quasi biclique characteristics are given inside the partition boxes. On the top left we get information for the currently examined node (the one with the cursor on it).

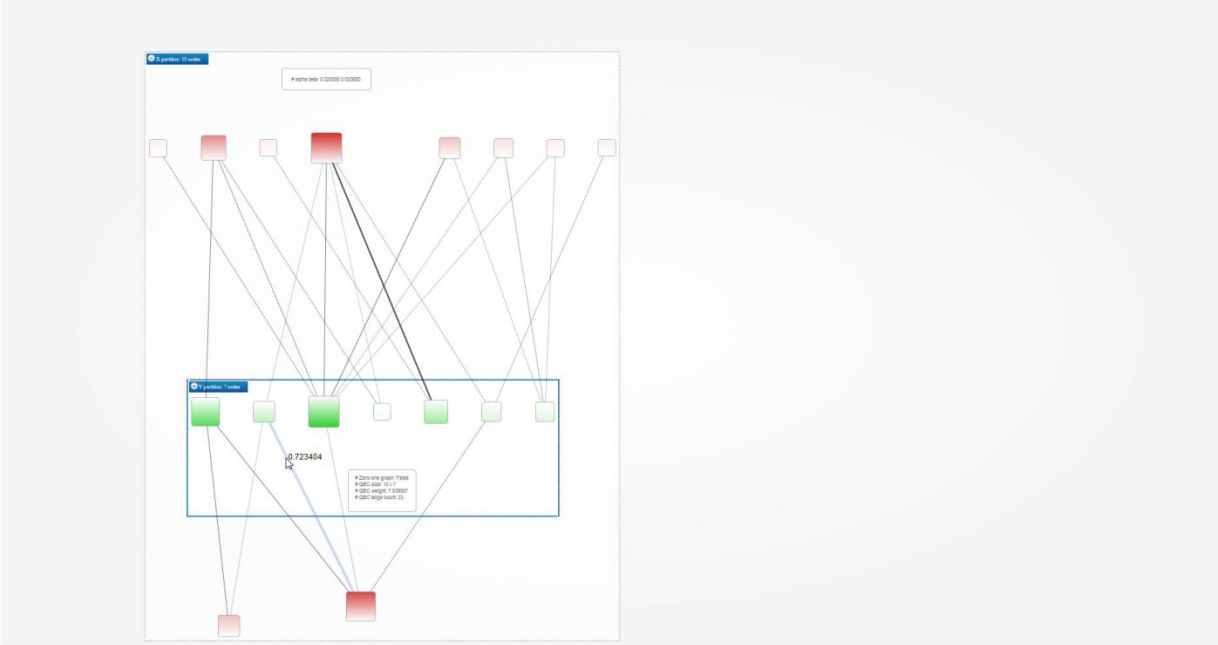


Figure 2.2.2. The graph is slightly zoomed out, a couple nodes are rearranged for better evaluation, and we are given the weight of the edge currently inspected (the one under the cursor).

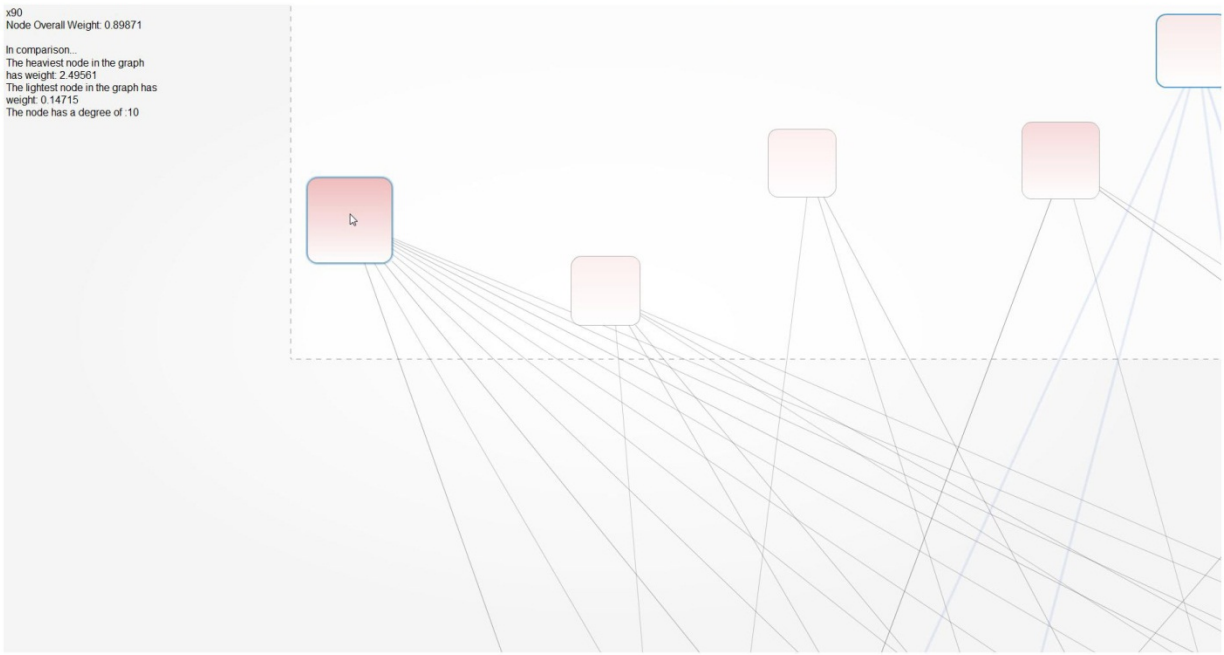


Figure 2.2.3. Zoomed-in view of a larger returned quasi biclique.

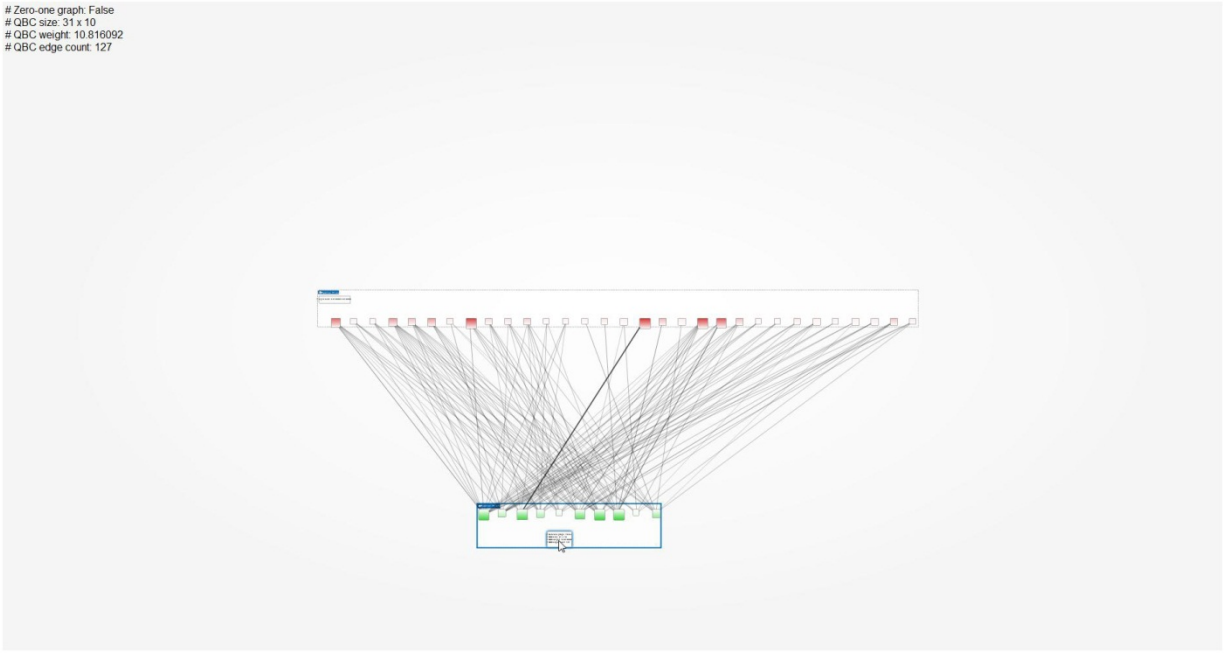


Figure 2.2.4. Zoomed-out view of a larger returned quasi biclique.

Chapter 3- Skeleton based heuristic

3.1 Motivation for the skeleton based heuristic

The application described in Chapter 4 gives exact solutions, however similarly to all other exact solutions, it is slow. Graphs of moderate size (e.g. 250 to 300 vertices) can take time well in excess of 15 to 20 minutes in order to calculate just one quasi biclique given certain parameters. For example, suppose we want to calculate up to ten different maximal quasi bicliques (and also assume that we can always find at least ten of those) for each different configuration of α β values, i.e. if we have an initial α=β=0.005 value and we want to move up to α=β=0.02 with an increment of 0.001 in each step, we would need for a moderately sized graph ~15 to 20 minutes for the first qbc obtained with α=β=0.005, another ~15 to 20 minutes for the second qbc obtained with α=β=0.005, another ~15 to 20 minutes for the third qbc obtained with α=β=0.005, and so on. Once we get the first ten, we would need another ~15 to 20 minutes for each of the ten qbcs that we will acquire with parameters α=β=0.006, then another ~15 to 20 minutes for each of the ten qbcs that we will acquire with parameters α=β=0.007, and so on. In other words, in practice it is not viable to ask for more than one qbc returned for each different αβ values configuration. Even then if the increment step is too fine we will need a long time to process the whole job. This however is a general flaw of all exact solution approaches and not one of the αβ weighted quasi biclique approach. It seems clear that what we need is fast heuristics that will sacrifice exactness on the altar of usability.

My proposed skeleton oriented heuristic is based in the following intuition: No matter what sort of quasi biclique definition we want to use, it better be the case that the parts of the original input that are densely enough populated by edges are participants in the returned quasi biclique. Obviously the most densely edge-populated

regions of a bipartite graph are its biclique subgraphs. If then we somehow had all the biclique (or even sufficiently many) subgraphs of a graph G we could try to construct a quasi biclique skeleton that contains as many of them as possible and check whether that skeleton satisfies our any-given definition of quasi-ness. Such an approach has the two disadvantages mentioned in 3.3 where we examined the α-quasi biclique biclique approach. Namely, it presupposes that we have all the bicliques of a bipartite graph and furthermore it presupposes that such bicliques always exist.

However in many cases it is either easy to calculate the cliques of a graph or we already have done so in the past (for whatever reason) in which case we have the extracted cliques available anyway. For example, if we are to use the algorithm described in Alexe *et al*. for the extraction of cliques we would get results in polynomial time if our bipartite graph happens to be convex [33]. Similarly we get linear or polynomial time if the input graph happens to be among one of the special bipartite cases listed in [34], for example 1-bounded or 2-bounded bipartite [35]. Furthermore, although in my description I will be talking exclusively about bicliques, the theory will work even when we relax the requirement of using bicliques for constructing the skeleton, requiring instead locally dense enough areas or local qbcs (not necessarily maximal).

3.2 Skeleton preliminaries

Let $G := (U \cup V, E)$ be a bipartite graph and let $U_X \subseteq U, V_Y \subseteq V$.

Let $U_{XY} \subseteq U_X$ such that $x \in U_{XY} \Leftrightarrow \exists y \in V_Y \ s.t \ (x, y) \in E$.

Similarly:

Let $V_{YX} \subseteq V_Y$ such that $x \in V_{YX} \Leftrightarrow \exists y \in U_X \ s.t \ (x, y) \in E$

Let A, B be two complete subgraphs of G such that A is the biclique subgraph of G induced by (U_A, V_A) and B is the biclique subgraph of G induced by (U_B, V_B).

We define the **degree of A to B** as $\deg(A \to B) = |U_{AB}| + |V_{AB}|$

We define the **degree of the pair AB** as $\deg(A \leftrightarrow B) = \deg(A \to B) + \deg(B \to A)$

We define the **density of A to B** as $den(A \to B) = \dfrac{\deg(A \to B)}{|A|}$

We define the **density of the pair AB** as $den(A \leftrightarrow B) = \dfrac{\deg(A \leftrightarrow B)}{|A| + |B|}$

For example in Figures 3.2.1 a and b we get:

$\deg(A \to B) = 3 + 3 = 6$

$\deg(B \to A) = 1 + 2 = 3$

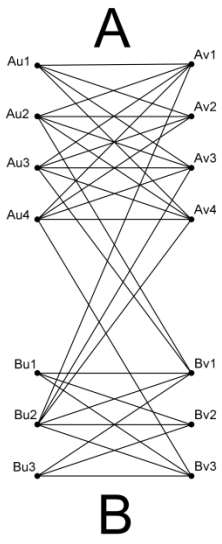
$\deg(A \leftrightarrow B) = 6 + 3 = 9$

$den(A \to B) = \dfrac{6}{8}$

$den(B \to A) = \dfrac{3}{6}$

$den(A \leftrightarrow B) = \dfrac{9}{8+6} = \dfrac{9}{14}$

Obviously a density of 1 implies that A and B are sub-cliques of a biclique.



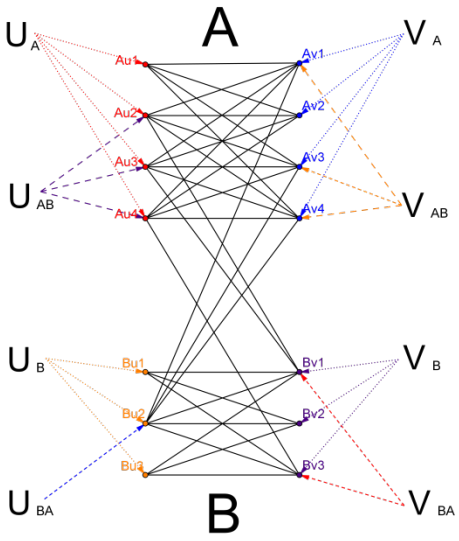


Figure 3.2.1a (top) An example with two bicliques A and B. **3.2.1b** (bottom) The same example with the nodes colored and the sets of interest identified.

Let $S = \{S_1, S_2, ..., S_n\}$ be a set of n complete subgraphs of G.

Definition 3.2.1: *skeleton*

We say that $Sk \subseteq S$ is a **weak skeleton** or **d-skeleton** or simply **skeleton** if for all $\forall S_i \in Sk,\ \exists S_j \in Sk$ s.t. $\deg(S_i \to S_j) \geq d$ or $\deg(S_j \to S_i) \geq d$ where d is a positive integer.

Definition 3.2.2: *strong skeleton*

We say that $Sk \subseteq S$ is a **strong skeleton** or a **strong d-skeleton** if for all $\forall S_i \in Sk,\ \exists S_j \in Sk$ s.t. $\deg(S_i \leftrightarrow S_j) \geq d$ 3here d is a positive integer.

Definition 3.2.3: *density- skeleton*

We say that $Sk \subseteq S$ is a **weak density-skeleton** or **density-r-skeleton** or simply **density-skeleton** if for all $\forall S_i \in Sk,\ \exists S_j \in Sk$ s.t. $den(S_i \to S_j) \geq r$ or $den(S_j \to S_i) \geq r$ where $0 < r \leq 1.$

Definition 3.2.3: *strong density- skeleton*

We say that $Sk \subseteq S$ is a **strong density-skeleton** or **strong density-r-skeleton** if for all $\forall S_i \in Sk,\ \exists S_j \in Sk$ s.t. $den(S_i \leftrightarrow S_j) \geq r$ where $0 < r \leq 1.$

Definition 3.2.4: *skeleton weight*

The **weight of a skeleton** is defined as :

$$\sum_{S_i, S_j \in Sk} \deg(S_i \to S_j)$$

Definition 3.2.5: *strong skeleton weight*

The **weight of a strong skeleton** is defined as :

$$\sum_{S_i, S_j \in Sk} \deg(S_i \leftrightarrow S_j)$$

Similarly we define the weight of a density-skeleton and a strong density skeleton.

Definition 3.2.6: *maximum weight skeleton*

A **maximum weight skeleton** (or maximum weight density-skeleton) is a skeleton (or density skeleton) with a weight at least as much as the weight of any other skeleton (or density skeleton) in S.

Problem 3.2.1: *Maximum Skeleton Extraction*

Instance: Given a set $S = \{S_1, S_2, ..., S_n\}$ of n complete subgraphs of a bipartite graph G and a positive integer d.

Query: Find the maximum weak (or strong) weight skeleton in S and report its weight.

The Maximum Density Skeleton Extraction problem is defined similarly.

3.3 The algorithm

The problem of Maximum Skeleton Extraction (all versions, i.e. weak, strong, weak density, strong density) from an input bipartite graph G given that we have the set S can be solved in polynomial time with the algorithm described below. In this example we will assume that we are looking for a Maximum Strong Skeleton.

1. First we construct a new graph G' as follows:
   a) For each $S_i \in S$ with $i \in [1, n]$ we add a new node i in G'.
   b) Two nodes u and v in G' are connected with an undirected edge $\{u, v\}$ with a weight of $\deg(S_i \leftrightarrow S_j)$ if and only if S_u and S_v satisfy the strong skeleton condition i.e. $\deg(S_i \leftrightarrow S_j) \geq d$.

2. We run a DFS on the resulting graph G' and we acquire all its connected components.

3. For each of the connected components $Cc_i \in G'$ we acquire its weight: $\sum_{S_i, S_j \in Cc_i} \deg(S_i \leftrightarrow S_j)$.

4. We then acquire a subset vertex set $(U \cup V)_{Sub} \subseteq (U \cup V)$ of the original vertex set of graph G as follows:
    a) For each vertex u in the heaviest component of G' we add in $(U \cup V)_{Sub}$ all the vertices contained in the corresponding biclique $(U \cup V)_{S_u}$.

5. We return the subgraph of G induced by the vertex set $(U \cup V)_{Sub}$ as our result.

It is easy to check that the above algorithm needs only polynomial time to run. Now that we have a maximum skeleton we can check again in polynomial time whether it is actually good enough according to our criteria as to be considered a quasi biclique.

Problem 3.3.1: *QBC status check*

Instance: Given a maximum skeleton $mSkel$ of a bipartite graph G and some quasi-ness criterion (e.g. αβ-weighted quasi biclique with $w: E \to [0,1]$).

Query: Is $mSkel$ a quasi biclique according to our criterion? (e.g. is it an αβ-weighted quasi biclique with $w: E \to [0,1]$?)

We can check the answer to the above question in polynomial time with the following algorithm inspired by the survey design paradigm described in [36]. (I will keep counting the steps as if we are continuing immediately from step 4 of the maximum skeleton extraction as the question posed in 3.2.2 is a direct extension of the question posed in 3.2.1.)

6. First we construct a new graph G_f as follows:

   a) For each node in $mSkel$ we add a new node in G_f.

   b) For each edge in $mSkel$ we add the corresponding edge in G_f with 0,1 capacity and we direct them from the nodes in $U(mSkel)$ to the nodes in $V(mSkel)$.

   c) We then add two new nodes: a source and a sink.

   d) From the source node we bring a directed edge to each of the nodes in $U(mSkel)$ with low and high capacity depending on our criterion. For example if our criterion is that each of our nodes in $U(mSkel)$ should be connected to at least c_l but no more than c_h nodes in $V(mSkel)$ then our low capacity will be c_l and our high capacity will be c_h where $0 \le c_l \le c_h \le |V(mSkel)|$. Of course we do not have to restrict the upper bound in which case the upper bound would be simply $|V(mSkel)|$ since we cannot have a node in $U(mSkel)$ having a degree higher than $|V(mSkel)|$.

e) From each node in $|V(mSkel)|$ we bring a directed edge to the sink node with low and high capacity again depending on our criterion. For example if we want each node in $V(mSkel)$ to be connected to at least d_l but no more than d_h nodes in $U(mSkel)$ then our low capacity will be d_l and our high capacity will be d_h where $0 \le d_l \le d_h \le |U(mSkel)|$. Once again we do not have to restrict the upper capacity in which case we would simply have $|U(mSkel)|$ to be the upper bound.

f) Finally we bring an edge from the sink to the source with capacity $|U(mSkel)| \cdot c_h$.

7. We then ask whether there exists a feasible circulation in the resulting graph G_f .

An example construction based on the graph of Figure 3.2.1 can be seen in Figure 3.3.1.

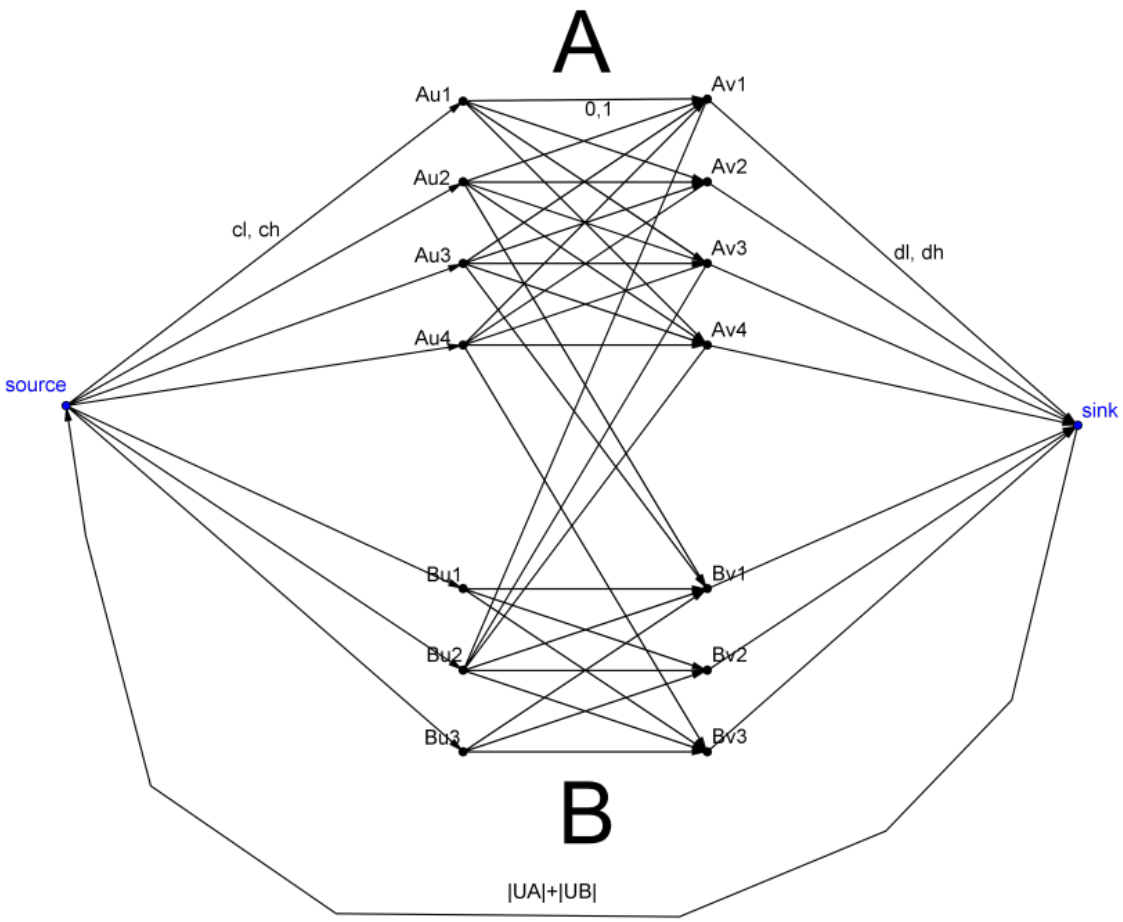


Figure 3.3.1. An example construction of G_f based on the example given in Figure 5.2.1.

If the answer is yes, then the $mSkel$ is a qbc according to our criterion. This is because if we have feasible circulation (and thus an integral circulation) that means that there is a way to satisfy the condition that each node in $U(mSkel)$ can send at least c_l units of flow to $V(mSkel)$ and each node in $V(mSkel)$ can receive at least d_l units of flow from nodes in $U(mSkel)$ (and the circulation simply shows how this is possible) which is precicely the way we set up the criterion of quasi-ness to be by construction. The converse is also true. If the $mSkel$ is a qbc according to our criterion then each node in $U(mSkel)$ has to have at least c_l neighbors in $V(mSkel)$ and each node in $V(mSkel)$ has to have at least d_l neighbors in $U(mSkel)$ and thus we can send the appropriate number of units of flow through each node. The weight of such qbc will be simply $|E(mSkel)|$ since we are ignoring the edge weights and we are concerned with unweighted qbcs.

We can then proceed in trying to see if we can actually extend our skeleton so that we increase the weight of our qbc. We could do so in several ways, for example by sorting all the vertices that are not in $mSkel$ but are the endpoint of an edge that has an endpoint in $mSkel$ according to their $mSkel$-degree (i.e. how many edges that touch nodes in $mSkel$ have them as their other end point) and adding them one by one to $mSkel$, checking each time whether the addition will break the quasi-ness or not (a greedy approach that will not necessarily guarantee maximality), or by using dynamic programing to choose those vertices. As another example we could use the resulting $mSkel$ as the basis for an extension as described in the α - quasi biclique section.

If the answer is negative then we answer that the current $mSkel$ is not a qbc according to our criterion and proceed by checking the next biggest skeleton extracted at step 3a, $mSkel'$, repeating the whole process . If we have exhausted our inputs (i.e. we have checked all the skeletons extracted by the connected components of G')

without having a positive result then we can answer that we could not find a qbc that satisfies the given criterion.

3.4 Algorithm discussion

One thing that makes this algorithm appealing (besides its polynomial time) is its flexibility and its ability to be parallelized. For example, one could check a skeleton for quasi-ness using several criteria at once using several threads, where each thread runs the check on G_f for a different criterion. We could check whether G_f has a feasible flow according to the γ-quasi biclique criterion in thread one while checking if it has a feasible flow according to the δ- quasi biclique criterion in thread two. In fact we could simultaneously use another two thread to check these two criteria for $mSkel'$, i.e. the second largest skeleton found, use another two threads to check simultaneously the same criteria for $mSkel''$, i.e. the third largest skeleton found, and so on.

The ability to use multiple criteria is not the only factor that makes the algorithm flexible. In fact we can set individual criteria for each of the participating nodes in our Skeleton by setting different minimum and maximum capacities for the edges from the source to $U(mSkel)$ for each $u \in U(mSkel)$ and for the edges from each $v \in V(mSkel)$ to the sink if, for example, we have vertices that we deem to be more or less important than the rest. We can also enforce balance and symmetry by manipulating the capacities appropriately (e.g. enforce symmetry by setting $c' = d'$) depending on whether that would be useful for our case.

Furthermore the algorithm gives us the tools needed to evaluate the quality of our qbcs according to different needs. One question that arises naturally once we have the results of a heuristic is how to evaluate and compare them to the exact solutions. In our case, should we consider simply the weight of the resulting qbc, should we compare

the structural similarities or should we consider something else still? For example, consider Figure 3.4.1. If all we care about is the weight of a qbc, then by requiring that the strong skeleton condition is $\deg(A \leftrightarrow B) \geq 6$ both graphs in 3.4.1 are good skeletons with :

$\deg(A \leftrightarrow B) = 6 + 3 = 9$ for the Case 1 graph and

$\deg(A \leftrightarrow B) = 5 + 1 = 6$ for the Case 2 graph.

i.e. both graphs qualify as skeletons but we would in fact prefer the graph in Case 2 as it will give a larger qbc-weight although the graph in Case 1 has a larger degree assuming that both graphs pass the quasi-ness test..

However, if we care about the structure of our skeleton as well, then by picking the strong density condition $1 \geq den(A \leftrightarrow B) \geq 1/2$ we get that the

$den(A \leftrightarrow B) = \frac{6+3}{8+6} = \frac{9}{14}$ for the Case 1 graph and

$den(A \leftrightarrow B) = \frac{5+1}{10+4} = \frac{6}{14}$ for the Case 2 graph

, i.e. although the Case 2 graph remains heavier this time it will fail to qualify since its structural density is rather weak (only one node in B is responsible for all the connections to nodes in A and conversely all nodes in A that have a connection in B have a connection to only a single node in B).

Finally, as I mentioned earlier, despite the fact that the algorithm was described in this thesis as requiring bicliques, there is nothing that restricts us from using other notions for constructing the skeletons. We could therefore use any other method that we would like in order to generate dense enough skeletons; For example, we could make use of the vast literature on the extraction of dense subgraphs from graphs, e.g.

check [37-41]. The algorithm described up until step 7 is implemented in Java and the code is available in sourceforge.org.

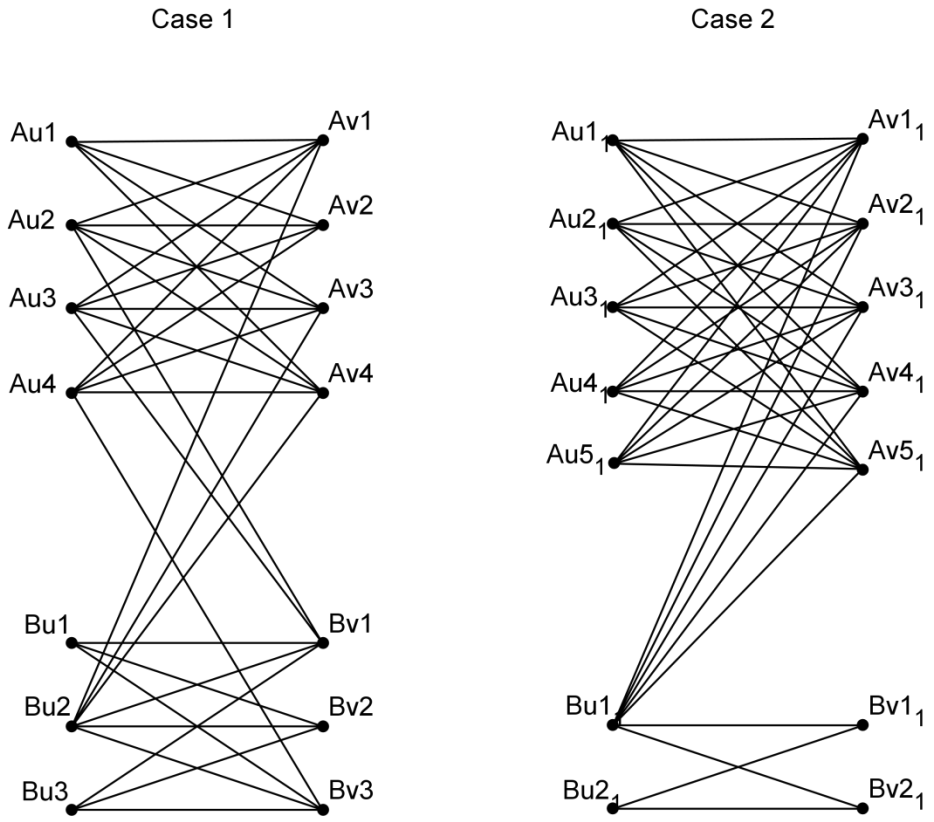


Figure 3.4.1. Example where different qbc evaluation criteria will result in different picks by our algorithm.

Chapter 4 - Future work

There are several possibilities for improvement of both the algorithm and its implementation that can be done in the future. A first step for improving the implementation would be to modify my code so that it will use multiple threads as opposed to being a single threaded application. Furthermore, several improvements can be made in its running time by re-touching the implementation specifics. For example, the algorithm used for max flow is the Edmonds – Karp which has running time $O(VE^2)$ [42]; however, the Dinic's algorithm can perform better using $O(V^2E)$ time[43]. A web interface similar to the one that was provided for the ILP formulation of the αβ-weighted quasi biclique problem can be provided for this application as well.

One of the major issues with the approach described is that it cannot handle the case where we are dealing with weighted bipartite graphs. An obvious improvement would be to extend the algorithm so that it can handle such graphs as well (one way to do that would be to use altered degree or density criteria based on the sum of the weight of the edges that connect any given two biclique subgraphs of the original graph and then alter the quasi-ness checker or use a new one).

Another step that should be taken next, is to use the algorithm on data given by experts and then have the experts evaluate the results. As it is now the evaluation of the usefulness of a qbc regardless of the algorithm or approach used to extract it can be done only by an expert who will judge the result. A comparison between the skeleton approach and some other approach, say γ-quasi bicliques is by nature vague since, as discussed earlier, there is more than one way to compare two qbcs (weight, structural similarity, etc.). Thus, in order to actually evaluate the results of the skeleton approach, one has first to decide what he or she is comparing the results to and what the important criterion for the comparison is. If, for example, we want to compare the

skeleton based result to a γ-qbc result and we care only about the weight of the result alone, we need to pick a density-based skeleton approach and find out what is the threshold that better approaches the weight given by the γ-qbc result. Only then we can make a statement about the quality of the returned result.

References

1. Chang Wen-Chieh, V.S., Krause Roland, Eulenstein Oliver, *Mining Biological Interaction Networks Using Weighted Quasi-Bicliques Bioinformatics Research and Applications*, J. Chen, J. Wang, and A. Zelikovsky, Editors. 2011, Springer Berlin / Heidelberg. p. 428-439.
2. W.Peng, C.D., T.Li and T. Sun, *Finding hotspots in document collection.* Proceedings of ICTAI, 2007.
3. T.Murata, *Discovery of user communities from web audience measurement data.* Proceedings of WI, 2005: p. 673-676.
4. A.Z. Broder, R.K., F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J.L. Wiener, *Graph Structure in the Web.* Computer Networks, 2000. **33**(nos. 1-6): p. 309-320.
5. R. Kumar, P.R., S. Rajagopalan, and A. Tomkins, *Trawling the Web for Emerging Cyber-Communities.* Computer Networks, 1999. **31**(no. 11-16): p. 1481-1493.
6. J.E. Rome, R.M.H., *Towards a Formal Concept Analysis Approach to Exploring Communities on the World Wide Web.* Proc. Int'l Conf. Formal Concept Analysis, 2005: p. 33-48.
7. A.C. Driskell, C.A., J.G.B.M.M. McMahon, B.C. OMeara, and M.J. Sanderson, *Prospects for Building the Tree of Life from Large Sequence Databases.* Science, 2004. **306**: p. 1172-1174.
8. M.J. Sanderson, A.C.D., R.H. Ree, O. Eulenstein, and S.Langley, *Obtaining Maximal Concatenated Phylogenetic Data Sets from Large Sequence Databases.* Molecular Biology and Evolution, 2003. **20**(no. 7): p. 1036-1042.
9. C. Yan, J.G.B., and O. Eulenstein, *Identifying Optimal Incomplete Phylogenetic Data Sets from Sequence Databases.* Molecular Phylogenetics and Evolution, 2005. **35**(no. 3): p. 528-535.
10. H. Li, J.L., and L. Wong, *Discovering Motif Pairs at Interaction Sites from Protein Sequences on a Proteome-Wide Scale.* Bioinformatics, 2006. **22**: p. 989-996.
11. Schwikowski, D.J.R.a.B., *Predicting Protein-Peptide Interactions via a Network-Based Motif Sampler.* Bioinformatics, 2004. **20**: p. i274-i282.
12. A.H. Tong, B.D., G. Nardelli, G.D. Bader, B. Brannetti, L. Castagnoli, M. Evangelista, S. Ferracuti, B. Nelson, S. Paoluzi, M. Quondam, A. Zucconi, C.W. Hogue, S. Fields, C. Boone, and G. Cesareni, *A Combined Experimental and Computational Strategy to Define Protein Interaction Networks for Peptide Recognition Modules.* Science, 2002. **295**: p. 321-324.
13. A. Thomas, R.C., N.A.M. Monk, and C. Cannings, *On the Structure of Protein-Protein Interaction Networks.* Biochemical Soc. Trans., 2003. **31**: p. 1491-1496.
14. J.L. Morrison, R.B., D.J. Higham, and D.R. Gilbert, *A Lock-and-Key Model for Protein-Protein Interactions.* Bioinformatics, 2006. **22**(no. 16): p. 2012-2019.
15. B. Andreopoulos, A.A., X. Wang, M. Faloutsos, and M. Schroeder, *Clustering by Common Friends Finds Locally Significant Proteins Mediating Modules.* Bioinformatics, 2007. **23**(no. 9): p. 1124-1131.

16. D. Bu, Y.Z., L. Cai, H. Xue, X. Zhu, H. Lu, J. Zhang, S. Sun, L. Ling, N. Zhang, G. Li, and R. Chen, *Topological Structure Analysis of the Protein-Protein Interaction Network in Budding Yeast.* Nucleic Acids Research, 2003. **31**(9): p. 2443-2450.
17. H. Hishigaki, K.N., T. Ono, A. Tanigami, and T. Takagi, *Assessment of Prediction Accuracy of Protein Function from Protein-Protein Interaction Data.* Yeast, 2001. **18**(no. 6): p. 523-531.
18. Yannakakis, M., *Node Deletion Problems on Bipartite Graphs.* SIAM J. Computing, 1981. **10**: p. 310-327.
19. Peeters, R., *The Maximum Edge Biclique Problem Is NPComplete.* Discrete Applied Math., 2003. **131**(no. 3): p. 651-654.
20. Liu, X.L., J., Wang, L., *Quasi-bicliques: Complexity and Binding Pairs. In Hu, X., Wang, J. (eds.) COCOON 2008.* LNCS, 2008. **5092**: p. 255-264.
21. Liu, X.L., J., Wang, L., *Modeling protein interacting groups by quasi-bicliques: complexity, algorithm, and application.* IEEE/ACM Trans Comput Biol Bioinform, 2010. **7**(2): p. 354-364.
22. J. Abello, M.G.C.R., and S. Sudarsky, *Massive quasi-clique detection.* LATIN 2002, LNCS 2286, 2002: p. 598–612.
23. S. Arora, C.L., R. Motwani, M. Sudan, and M. Szegedy, *Proof verification and hardness of approximation problems.* Proc. 33rd IEEE Symp. on Foundations of Computer Science, 1992: p. 14–23.
24. Safra, S.A.a.S., *Probabilistic checking of proofs: A new characterization of NP.* J. of the ACM, 1998. **45**: p. 70-122.
25. U. Feige, S.G., L. Lov´asz, S. Safra, and M. Szegedy, *Approximating the maximum clique is almost NP-complete. .* In Proc. 32nd IEEE Symp. on Foundations of Computer Science, 1991: p. 2-12.
26. H˚astad, J., *Clique is hard to approximate within n^(1-ε).* Acta Mathematica., 1999. **182**: p. 105-142.
27. Resende., T.A.F.a.M.G.C., *A probabilistic heuristic for a computationally difficult set covering problem.* Operations Research Letters, 1989. **8**: p. 67-71.
28. N. Mishra, D.R., and R. Swaminathan, , *A new conceptual clustering framework, .* J Mach Learn Res, 2005. **56(1-3)**: p. 115-151.
29. Sim, K., et al., *Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks.* Stat. Anal. Data Min., 2009. **2**(4): p. 255-273.
30. Karp, R.M., *Reducibility among Combinatorial Problems.* Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., 1972: p. 85-103.
31. D. Peleg, G.S., and A. Wool, *Approximating Bounded 0-1 Integer Linear Programs.* Proc. Second Israel Symp. Theory of Computing and Systems (ISTCS), 1993.
32. L.Wang, *Near Optimal Solutions for Maximum Quasi-bicliques.* M.T.Thai and S. Sahni (Eds): COCOON 2010, LNCS 6196, 2010: p. 409-418.
33. Alexe, G., et al., *Consensus algorithms for the generation of all maximal bicliques.* Discrete Applied Mathematics, 2004. **145**(1): p. 11-21.
34. H.N. de Ridder, e.a., *http://www.graphclasses.org/classes/problem_Clique.html part of the Information System on Graph Classes and their Inclusions (ISGCI)* 2001-2012 updated 20 June 2012.
35. Golumbic, M.C., *Algorithmic graph theory and perfect graphs.* 1980.

36. Kleinberg, J. and E. Tardos, *Algorithm Design*2005: Addison-Wesley Longman Publishing Co., Inc.
37. Andersen, R., *A local algorithm for finding dense subgraphs.* ACM Trans. Algorithms, 2010. **6**(4): p. 1-12.
38. Andersen, R. and K. Chellapilla, *Finding Dense Subgraphs with Size Bounds*, in *Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph*2009, Springer-Verlag: Barcelona, Spain. p. 25-37.
39. Charikar, M., *Greedy approximation algorithms for finding dense components in a graph*, in *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*2000, Springer-Verlag. p. 84-95.
40. Feige, U. and M. Seltser, *On the densest k-subgraph problems*, 1997, Weizmann Science Press of Israel.
41. Gibson, D., R. Kumar, and A. Tomkins, *Discovering large dense subgraphs in massive graphs*, in *Proceedings of the 31st international conference on Very large data bases*2005, VLDB Endowment: Trondheim, Norway. p. 721-732.
42. Edmonds, J. and R.M. Karp, *Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems.* J. ACM, 1972. **19**(2): p. 248-264.
43. Dinic, E.A., *Algorithm for Solution of a Problem of Maximum Flow in a Network with Power Estimation.* Soviet Math Doklady, 1970. **11**: p. 1277-1280.

Acknowledgements and Dedication

I would like to thank my major professor, Dr. Oliver Eulenstein, for giving me the opportunity to work with him for my Master's degree. His support and advice was always useful and much appreciated.

I would also like to thank Drs. Fernandez-Baca and Slutzki whose classes inspired me to pursue graduate study.

Finally, I would like to thank Drs. Martin and Bauman for serving on my committee.

This work is dedicated to my wife, Allison, and my three month old daughter, Athena, whose arrival made the completion of this thesis considerably more challenging (and my life considerably more beautiful).