Searching Maximum Quasi-Bicliques

1 Theoretical background

1.1 Preliminaries

Let us introduce a couple of notions.

A graph G is called a *bipartite graph* if its vertex set can be partitioned into two sets, say U and V such that every edge of G has one point in U and other point in V. The set $\{U, V\}$ is called a bipartition of G. We denote such a bipartite graph by $G = (U \bigcup V, E)$, where E is the set of edges of G.

A complete subgraph of a graph G is called a *clique*. The MAXIMUM CLIQUE PROBLEM (**MCP**) is to find a complete subgraph of maximum cardinality in a general graph. A complete bipartite subgraph in a bipartite graph $G = (U \bigcup V, E)$ is called a *biclique*. The MAXIMUM VERTEX BICLIQUE (**MVB**) problem is to find a biclique of G with maximum number of vertices. The MVB problem is polynomial time solvable for bipartite graph. The MAXIMUM EDGE BICLIQUE (**MEB**) problem is to find a biclique of G with maximum number of edges. The decision version of **MEB** remains NP-complete even for bipartite graphs. Here, we are interested in variations of these problems where the search is conducted for quasi-bicliques (i.e. the constraint for searching a complete subgraph is relaxed).

In a graph $G = (V, E)$ a subgraph $G' = (V', E')$, where $V' \subseteq V, E' \subseteq E$, is called a *vertex-induced subgraph*. We denote such graph as $G[V']$. The *density* of an arbitrary graph is the ratio of the number of edges to the maximum possible number of edges. The density of a bipartite graph $G = (U \bigcup V, E)$ is $\rho = \frac{|E|}{|U| \times |V|}$. A γ-*quasi-biclique* in a bipartite graph $G = (U \bigcup V, E)$ is its bipartite induced subgraph $G' = (V' \bigcup U', E' \subseteq U' \times V')$ with the density at least $\gamma \in (0, 1]$.

The MAXIMUM VERTEX QUASI-BICLIQUE (**MVQB**) problem in a bipartite graph $G = (U \bigcup V, E)$ with fixed $\gamma \in (0, 1]$ is to find $V' \subseteq V, U' \subseteq U$, such that vertex-induced subgraph $G[U' \bigcup V']$ is a γ-quasi-biclique of size $|U'| + |V'|$, maximum for this graph.

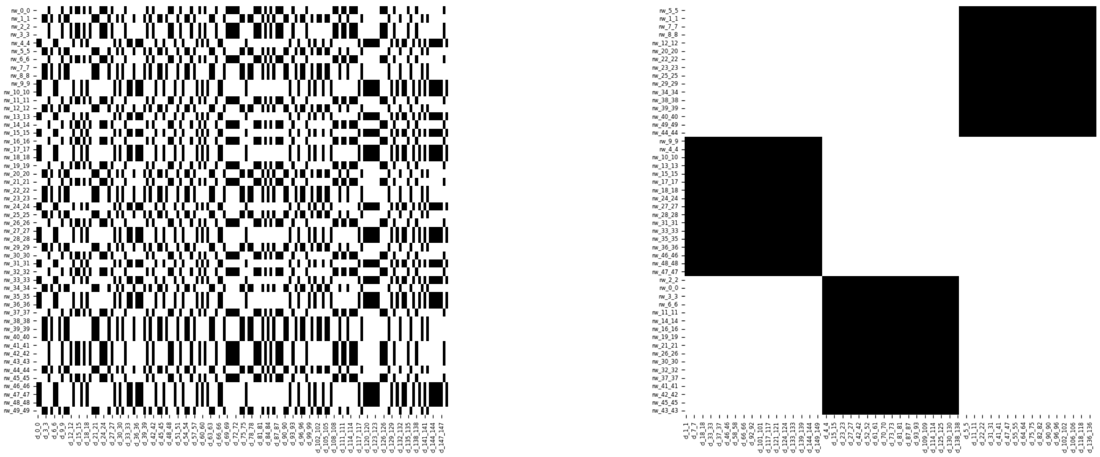
The MAXIMUM EDGE QUASI-BICLIQUE (**MEQB**) problem in a bipartite graph $G = (U \bigcup V, E)$ with fixed $\gamma \in (0, 1]$ is to find $V' \subseteq V, U' \subseteq U$, such that the vertex-induced subgraph $G = (U' \bigcup V', E' \subseteq U' \times V')$ is a γ-quasi-biclique that maximizes the size of the set E'.

1.2 Application to maximum biclustering problem

Biclustering is a data mining technique used to analyze large data-sets with the goal of identifying groups of objects that exhibit similar behavior across a subset of attributes. The output of a biclustering algorithm is a set of sub-matrices, each representing a group of objects that are highly correlated across a subset of attributes. Given a matrix $A \in Z_2^{|U| \times |V|}$ with coefficients being 0 or 1 and a set of rows U(resp. columns V), we search for the largest sub-matrix containing mostly 1 coefficients (small percentage of errors (i.e. 0 coefficients) is acceptable). See for illustration Figures 1 and 2.

To address the aforementioned issue, we introduce a bipartite graph $G = (U \bigcup V, E \subseteq U \times V)$ where the set U(resp. V) corresponds to the set of rows(resp. columns) of a matrix. The edges $e \in E$ are associated with coefficients of value 0 in the matrix. Each vertex v is provided with weights, $deg(v)$ and $\overline{deg(v)}$ that correspond to the number of 1(resp. 0) in the associated row(resp. column). We relate binary variables x_{ij}, u_i and v_j to edge e_{ij}, row i and column j, respectively. A binary variable equals 1 to indicate the selection of the corresponding edge, row, or column, and 0 otherwise.

The bellow models are proposed to solve various bi-clustering problems.



(a) An instance of input data

(b) Three clusters have been found

Figure 1: Clustering a matrix without errors

1.2.1 Maximization-based variants for the MEQB and MVQB problems

Here we use binary variables x_{ij}, u_i and v_j to denote matrix cell i, j, row i, and column j selection, respectively. A binary variable equals 1 to indicate that the associated cell, row, or column, **remains** in the selected submatrix, and 0 otherwise (i.e. **deleting** it from the chosen submatrix.)

The following linear program is used in the model:

$$\max \sum_{i \in U} \sum_{j \in V} A_{i,j} x_{ij}, \tag{1}$$

$$x_{ij} \leq v_i, \ \forall i \in U, \forall j \in V \tag{2}$$

$$x_{ij} \leq v_j, \ \forall i \in U, \forall j \in V \tag{3}$$

$$x_{ij} \geq u_i + v_j - 1, \ \forall i \in U, \forall j \in V \tag{4}$$

$$\sum_{i \in U} \sum_{j \in V} (1 - A_{i,j}) x_{ij} \leq (1 - \gamma) \times \sum_{i \in U} \sum_{j \in V} x_{ij} \tag{5}$$

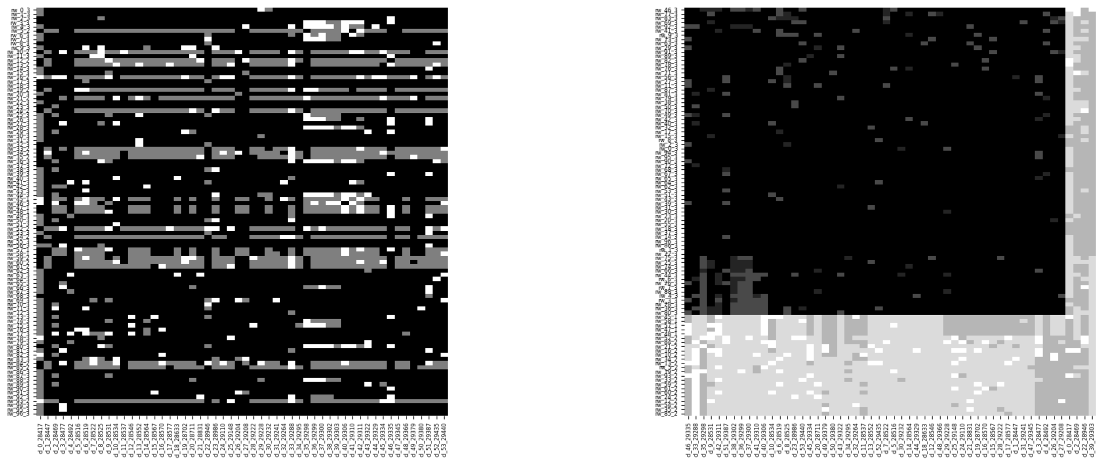
$$u_i, v_j \in \{0, 1\}, \ x_{ij} \in \{0, 1\} \ \forall i \in U, \ \forall j \in V \tag{6}$$

The function to maximize, (1), counts for the number of ones in a chosen submatrix determined by the binary variables having value 1. Constraints (2), (3) and (4) mean that matrix cell i, j is selected into the solution (i.e. $x_{ij} = 1$) if and only if its corresponding row i and column j are also chosen into the solution (i.e. $u_i = 1$ and $v_j = 1$).

The coefficient A_{ij} represents the value of the cell at position i and j, when searching for occurrences of 1s in the matrix, A_{ij} is used directly. However, if the search is for 0s, the coefficient is reversed to $(1 - A_{ij})$. Hence, constraint (5) ensures that the number of zeros in the selected matrix (the left hand side in (5)) is no more than $(1 - \gamma)$ from the chosen matrix size (the right hand side of the constraint). We denote the above model as **MaxM1**.

We study two more variants of **MaxM1**. In the first one, **MaxM2**, (1) is replaced by

$$\max \sum_{i \in U} \sum_{j \in V} x_{ij} \tag{7}$$



(a) Another instance of input data

(b) A big bicluster with small errors.

Figure 2: Clustering accepting small errors

while in the second one, denoted as **MaxM3**, the objective (1) is substituted by

$$\max \sum_{i \in U} u_i + \sum_{j \in V} v_j. \tag{8}$$

The constraints (2), (3), (4) and (6) are common for the three models **MaxM1**, **MaxM2** and **MaxM3**. It should be noted that the goal of **MaxM2** is to maximize the surface of the submatrix that is searched, whereas **MaxM3** aims to maximize the total of the rows and columns of the targeted submatrix. Thus, **MaxM1** and **MaxM2** models have connection with the problem **MEQB**, while **MaxM3** solves the **MVQB** problem.

1.2.2 Minimization-based variants for the MVB and MEB problem

This approach is inspired from the famous König's theorem that claims: In a bipartite graph G the number of edges of a maximal cardinality matching is the same as the number of vertices in a minimum vertex cover of G.

Here, choosing a vertex v means **deleting** the associated row (or resp. column) from the matrix. Since edges correspond to coefficients 0 in the matrix A, according to constraint (9) and (10) each coefficient 0 is deleted in the remained submatrix.

$$u_i + v_j \geq 1, \ \forall (i,j) \in E, \tag{9}$$

$$u_i, v_j \in \{0,1\}, \ \forall i \in U, \ \forall j \in V \tag{10}$$

The above two constraints are common for the two models that we study here. In the first one, **MinDel_1**, the objective function (11) seeks to delete all zeros by eliminating the least amount of coefficients 1. The objective function (12) in the second model, **MinDel_RC**, aims to minimize the number of rows and columns when deleting all zeros in the remained matrix. Hence, model **MinDel_1** relates to **MEB** problem, while model **MinDel_RC** targets solving the **MVB** problem.

$$\min \sum_{i \in U} \deg(i) u_i + \sum_{j \in V} \deg(j) v_j \tag{11}$$

$$\min \sum_{i \in U} u_i + \sum_{j \in V} v_j \tag{12}$$

1.2.3 Minimisation-based variants for the MVQB and MEQB problem

This section extains the ideas from the previous one in the case of quasi-bicliques.

The meaning of the variables u_i and v_j is the same as above. The essential particularity is that here we admit that a small portion, say $\epsilon \times \rho$, where $0 \leq \epsilon \leq 0.5$, of the set of edges E (i.e. the set of 0 coefficients), remains in the chosen matrix. For this purpose we introduce binary variables $z_{ij}, \forall (ij) \in E$ where $z_{ij} = 1$ means edge $e_{i,j}$ is **deleted**, 0 otherwise. This is ensured by constraint (13) below. Note that $\sum_{(i,j)\in E} z_{ij}$ is the number of edges that are deleted. Therefore, constraint (14) guarantees that this number is sufficiently large and close to the size of E, (that is at least $(1 - \epsilon \times \rho) \times |E|$).

$$u_i + v_j \geq z_{ij},\ \forall (i,j) \in E \tag{13}$$

$$\sum_{(i,j)\in E} z_{ij} \geq (1 - \epsilon \times \rho) \times |E| \tag{14}$$

$$z_{ij} \in \{0,1\},\ \forall (i,j) \in E \tag{15}$$

$$u_i, v_j \in \{0,1\},\ \forall i \in U,\ \forall j \in V \tag{16}$$

Similarly to the previous section, we study two models that use the same constraints, (13)-(16), but differ by their objective functions that coincide with (11) and (12). The first one, **Q_MinDel_1**, applies the objective (11) to delete here almost all zeros by eliminating the least amount of coefficients 1. The second model, **Q_MinDel_RC**, uses the function (12) to minimize the number of rows and columns when deleting almost all zeros in the remained matrix. Hence, model **Q_MinDel_1** relates to **MEQB** problem, while model **Q_MinDel_RC** has connection with **MVQB** problem.

0-1 Knapsack heuristics In this paragraph we study the behavior of a heuristics for solving **MEQB** problem. The model, denoted here as **KP_QB**, employes variables u_i and v_j as before, jointly with the objective (11). However, it substituts constraints (13)-(15) by the below single knapsack constraint

$$\sum_{i\in U} \overline{deg(i)} u_i + \sum_{j\in V} \overline{deg(j)} v_j \geq (1 - \epsilon \times \rho) \times |E|. \tag{17}$$

2 Tasks to be done

2.1 General guidelines

This project contains two parts; the first part includes the implementation of several IP models and is common for all students. The second part is individual with different questions for each student. Some general guidelines are to be respected:

1. The project should be accomplished either alone or in pairs (binome). Identical codes and/or reports will not be accepted.
2. Python code to be returned should run without errors. Any code that does not run correctly will be considered as false;
3. The project must be submitted by e-mail on the due date. The e-mail must contain:
   (a) In object: **[M1-MIAGE(EIT-DSC)][Project] Prénom Nom**;
   (b) In attachment: a zip format named: **project-Prenom-NOM.zip**. This archive should contain all the elements of the project : data, Python code and report in pdf format. The report should contain the below tables with your results as well your comments and analyse concerning the behaviour of the implemented models.
4. For any question, email to rumen.andonov@irisa.fr.

FINAL DEADLINE: 8AM 11/12/2023

2.2 Data extraction

You will be given a list of input data containing various instances (synthetic as well extracted from public databases). These data files follow the CSV data format. This format is commonly used to model databases.

2.3 Implementation

You are supposed to implement in PULP the following models described in the theoretical section: **MinDel_1**, **MinDel_RC**, **MaxM1**, **MaxM2**, **MaxM3**, **Q_MinDel_1**, **Q_MinDel_RC** and **KP_QB**. Your program should require three arguments ; 1) name of the input file; 2) the model to be executed; 3) error rate value.

3 Individual Questions:

Individual questions will be given to you during the sessions on 12/12/2023 and 13/12/2023.

4 Expected results

The goal of this project is to run, compare and analyse the behavior of the implemented models on the set of provided instances. In order to achieve this, you will display the results in the tables that are suggested below. You will present the needed number of tables to illustrate the results for various values of the parameters γ and ϵ and for all data. The data in the column **size** will be a couple (row, col) where row (resp. col) corresponds to the number of rows(resp. columns) in your result (obtained selected submatrix). In the column MIP you will provide the relative MIP gap. In the case of maximization this gap equals $\frac{UB-LB}{UB}$ where UB stands for an upper bound, while LB stands for a lower bound (the value of a found feasible solution). It is recommended that the running time must be limited to two hours. The advantage of Branch&Bound approach is these values are available even when the program has been stopped because of running time limit.

Data	Input Graph Size				**MinDel_1**				**MinDel_RC**			
	$\|U\|$	$\|V\|$	$\|E\|$	ρ	time	size	ρ	MIP gap	time	size	ρ	MIP err.
D1												
D2												

Table 1: Results of maximum biclique search

Data	**MaxM1**				**MaxM2**				**MaxM3**			
	time	size	ρ	MIP err.	time	size	ρ	MIP gap	time	size	ρ	MIP err.
D1												
D2												

Table 2: Results of maximum γ-quasi-biclique with parameter $\gamma = 0.6$.

Data	**Q_MinDel_1**				**Q_MinDel_RC**				**KP_QB**			
	time	size	ρ	MIP err.	time	size	ρ	MIP gap	time	size	ρ	MIP err.
D1												
D2												

Table 3: Results of maximum quasi-biclique using minimization based optimization. Parameter $\epsilon =$???.