Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Exploring the complexity boundary of the Maximum Common Edge Subgraph problem

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Autor: Saveliy Vasiliev

Director: Dr. Javier Leonardo Marenco

Jurados: Dra. Flavia Bonomo y Dr. Min Chih Lin

March 19, 2013

Abstract

Dados dos grafos G y H con igual cantidad de vértices, el problema de encontrar un subgrafo común a ambos grafos que maximize la cantidad de aristas se denomina *problema de subgrafo máximo por aristas* (MCESP). Una definición equivalente es encontrar una asignación $f : V_G \longrightarrow V_H$ uno a uno que maximize la cardinalidad de f, donde la cardinalidad se define como $|\{uv \in E_G : f(u)f(v) \in E_H\}|$. Se sabe que este problema es $\mathcal{NP}$-completo en el caso general.

La mayoría de los trabajos relacionados con el MCESP hasta el día de hoy se centraron en obtener algoritmos eficientes para computar asignaciones de cardinalidades aceptables. Dado el escaso estudio de la complejidad del MCESP en la literatura nos proponemos contribuir a dicha área.

En este trabajo exploramos el comportamiento de la complejidad del MCESP cuando los grafos de entrada son restringidos a diversas familias, centrándonos en distinguir los casos $\mathcal{NP}$-completos de los polinomiales. Algunas de las familias estudiadas son grafos bipartitos, split, de intervalos, cografos, árboles y grillas. Por otro lado relacionamos el MCESP con el problema de *isomorfismo de grafos* y la clase de complejidad $\mathcal{GI}$. Por último estudiamos aspectos generales del comportamiento de las asignaciones y asignaciones óptimas.

Abstract

Given two graphs G and H with the same number of vertices, the problem of finding a common subgraph of both graphs that maximizes the number of edges is called *maximum common-edge subgraph problem* (MCESP). An equivalent definition is to find a 1-1 mapping $f : V_G \longrightarrow V_H$ that maximizes the cardinality of f, where the cardinality is defined as $|\{uv \in E_G : f(u)f(v) \in E_H\}|$. This problem is known to be $\mathcal{NP}$-complete in the general case.

Most of the works related with MCESP up to day were aimed to obtain efficient algorithms for computing mappings with reasonable cardinality. Given the limited study of the MCESP complexity in the literature, we propose ourselves to contribute to this area.

In this work we explore the behavior of MCESP complexity when the input graphs are restricted to diverse families, focusing on distinguishing the $\mathcal{NP}$-complete cases from polynomial cases. Some of the studied families are bipartite, split, interval, cographs, trees and grid graphs. On the other hand we relate the MCESP with the *graph isomorphism* problem and the $\mathcal{GI}$ complexity class. Finally we study general behavioral aspects of mappings and optimal mappings.

Acknowledgments

First of all I want to thank to my advisor Javier Marenco for the time and dedication devoted to this work. He provided me with many useful advices and guided me throughout the whole research process, contributing to an outstanding learning experience. Aside of this work he answered an enormous amount of questions about his life experiences, which aided me to take the decision of applying to a Ph.D. I think during this work a great friendship was made. I am really grateful to Flavia Bonomo and Min Chih Lin for accepting to be the jurys of this work.

During the last few years I met a lot of great people which all influenced in my undergraduate lifetime. Many of them contributed to my education and helped in taking important decisions, in particular Santiago Figueira, Min Chih Lin, Flavia Bonomo, Esteban Mocskos, Nahuel Olaiz, Veronica Becher, Diego Delle Donne, Francisco Soulignac, Marina Groshaus, Mariano Perez Rodriguez, Diego Fernandez Slezak, Joos Heintz, Esteban Feuerstein and Santiago Ceria.

On the other hand I met a lot of great classmates and shared lots of experiences with them. I am sure I will miss someone as above, but I will make my best to remember as many as possible. Thanks to Ivan Postolski, Marcelo Bianchetti, Federico Lebron, Alejandro Masserolli, Joaquin Rinaudo, Michel Mizrahi, Agustin Montero, Gabriel Senno, Gaston Bengolea Monzon, Jennifer Hughes, Ariel Camblor, Juan M. Martinez Camaño, Martin Miguel, Pablo Gauna and Leandro Lera Romero.

I want to give special thanks to the University of Buenos Aires and all the people that makes this institution work. It is really valuable that Argentina has some free of charge universities available for everyone, this is a very effective social lift for many people who cannot afford private education. Being a foreigner and receiving my education here makes me even more thankful.

Also I want to thank my friends Juan, Cristian, Douglas, Andres, Matias, Alejandro, Nicolas and Kevin for the support and critics in every decision I made, and for these nice years we shared. I am feeling really lucky to have such a group of friends.

I am really grateful to my love Daniela Rubial for sharing with me the last 5 years, for the great understanding of each other we have and the amazing relation we built.

Finally I want to give special thanks to my parents who supported me since I was born in all decisions I made, for all the suggestions and feedback they gave me throughout these years, and for the great effort they made to give me the possibility of devoting myself entierly to complete the undergraduate.

Contents

CHAPTER 1

Getting the grip

1.1. Introduction

An *optimization problem* defines a set of feasible solutions for each instance and an evaluation function that assigns a value to each solution, generally the goal is to maximize or minimize this value. Many optimization problems may be solved without computing power, for example, we can use calculus to maximize or minimize a differentiable function. Of course this is not always the case, the hypothesis required for calculus are very restrictive for many problems, for instance many problems involve discrete variables or structures, such as integer numbers, graphs or schedules - these are called *combinatorial optimization problems.*

Many of combinatorial optimization problems were found throughout the human history and few of them were solved by hand. With the introduction of computing power many of these problems were solved using existing or new algorithms, nevertheless some of these problems turned out to be really difficult even with the most advanced computing systems at the time. Different approaches were used to give at least some reasonable solution to these problems, such as heuristic or randomized algorithms.

A notion of time complexity was developed in the 70s, suprisingly enough this notion seems to capture exactly the “difficult” problems. These ideas were formalized using the Turing Machine computing model and became powerful analysis tools, this way the $\mathcal{NP}$*-completeness* theory was founded. Nowadays when facing a new combinatorial optimization problem one uses these techniques to get a better understanding of how difficult a problem may be, using this analysis one may define a more suitable strategy for solving the problem. Although many complexity notions were introduced since the $\mathcal{NP}$*-completeness* theory, this is still one of the most widely used.

Naturally a new trend in the scientific community appeared, namely to classify the problems in “difficult” and “not difficult”. Many results are classifications of restrictions of one problem, where a *restriction* of a combinatorial problem is a combinatorial problem with the same evaluation function, and where the set of feasible solutions is a subset of the original feasible solutions set. Our goal in this work is to contribute to such classification of one problem in the graph theory field.

A *graph* is basically a set of vertices with edges connecting them. Many combinatorial optimization problems may be formulated as a query to a graph, which is constructed using a given instance of the problem. Sometimes those graphs to be queried have certain interesting properties, these properties define a graph class. Every class has its own properties, and these can be exploited in the research of faster algorithms or, on the negative side, stating that a query on such graph class is “difficult”. This is one of the main motivations to study graph theory and explore the complexity of different queries on different graph families. In this work we will usually skip the real problem and go straight to well know graph classes.

A *subgraph* of a graph is a graph that can be obtained after removing some vertices and edges from the original graph. A very studied problem is to decide whether a graph is a subgraph of another graph, this is known to be a “difficult” problem. One possible generalization of this problem is, given two graphs, find the largest common subgraph, where we measure its size as the edge count, this generalization is also called the maximum common edge subgraph problem. This problem was first introduced by Bokhari in [4], where he studied how to assign modules to array processors in order to reduce the communication time between these modules. In [12] a polyhedral approach was explored to this problem. In these works some complexity results were given, but a classification was not the main goal. To our best knowledge this is the first complexity classification oriented work.

1.2. Preliminaries

1.2.1. Mathematical notation used in this work. Given A, B sets we define $A \subseteq B$ if each element from A also belongs to B, we define $A = B$ if $A \subseteq B$ and $B \subseteq A$. If $A \subseteq B$ and $B \neq A$ we note $A \subset B$. We define $A \cup B = \{x : x \in A \text{ or } x \in B\}$, $A \cap B = \{x : x \in A \text{ and } x \in B\}$, finally we note $A \dot{\cup} B$ to the set $A \cup B$ if $A \cap B = \emptyset$. We denote $A \times B = \{(a, b) : a \in A \text{ and } b \in B\}$. We denote $A \setminus B = \{x : x \in A \text{ and } x \notin B\}$, and $A \triangle B = A \cup B \setminus (A \cap B)$. Given a universe $\mathcal{U}$ and a set $A \subseteq \mathcal{U}$ we note $\overline{A} = \{x : x \in \mathcal{U} \text{ and } x \notin A\}$, if obvious we omit the definition of $\mathcal{U}$. We note $\mathcal{P}(A) = \{B : B \subseteq A\}$. We say that $f \in \mathcal{P}(A \times B)$ is a *mapping* or *function* if for each $a \in A$ there is exactly one pair $(a, b) \in f$, we note this pair $f(a) = b$. If $f \in \mathcal{P}(A \times B)$ is a function we note $f : A \longrightarrow B$. Given $f : A \longrightarrow B$, if for each $b \in B$ there is exists exactly one $a \in A$ such that $f(a) = b$ we say that f is a *bijection* or *1-1 mapping*, and we note $f^{-1}(b) = a$. If $X \subseteq A$ and $f : A \longrightarrow B$ we note $f(X) := \{f(x) : x \in X\}$. Finally, given $f : A \longrightarrow B$ and $X \subseteq A$, we note $f\big|_X : X \longrightarrow B$ the function defined by $f\big|_X(x) = f(x)$ for each $x \in X$, and we call $f\big|_X$ the *restriction of f to X* or *f restricted to X*.

1.2.2. Basic graph theory. A *directed graph* is a pair $G := (V, E)$, where V is a finite set of *vertices* and $E \subseteq V \times V$ is the set of *edges*. We denote $m := |E|$ and $n := |V|$, if needed we add the subindex G to disambiguate. We write $V = V_G = V(G)$ and $E = E_G = E(G)$. If $(u, v) \in E$ we write $uv = (u, v)$ and we say that u and v are neighbors. A pair $G = (V, E)$ where $E \subseteq \{\{u, v\} : u, v \in V\}$ is called *undirected graph*. For $v \in V$ we define the *open neighbor set of v* as $N(v) := \{u \in V : uv \in E\}$, and the *degree* of v as $\deg(v) := |N(v)|$. If $G = (V, E)$, $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$ then (V', E') is a subgraph of G, if $E' = E \cap (V' \times V')$ then we say that (V', E') is an *induced* subgraph of G and we note $G[V'] := (V', E')$.

If G, H are graphs we define the *union* of G and H as $G \cup H := (V_G \dot{\cup} V_H, E_G \dot{\cup} E_H)$. The *join* of G and H, $G \oplus H$ is $G \cup H$ where each vertex of G is neighbor of each vertex of H. Finally the *cartesian product* of

G and H, written $G \times H$, is the graph $(V_G \times V_H, E)$ where $((u,x),(v,y))$ is an edge if and only if either $u = v$ and $xy \in E_H$, or $x = y$ and $uv \in E_G$.

Given a graph $G = (V, E)$, the *adjacency matrix* of G, $\mathrm{Adj}(G) \in \{0,1\}^{n \times n}$ is such that the entry ij is 1 if and only if there is an edge from v_i to v_j in the case of directed graphs, and simply an edge between i and j in the case of undirected graphs. Note that $\mathrm{Adj}(G)$ is a symmetric matrix in the undirected case.

A *complete graph* $K_n = (V, E)$ is a graph with $|V| = n$ such that each pair or different vertices are connected. Note that we have $n(n-1)/2 = |E|$.

A *cycle* is a graph $C_n = (V, E)$ with $n \geq 3$ and $V = \{v_1, \ldots, v_n\}$ such that $E = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n, v_nv_1\}$. Given G a graph a *circuit of length* k in G is a sequence of vertices $v_1, \ldots, v_k$ such that $v_iv_{i+1} \in E$ for $1 \leq i \leq k$ and $v_kv_1 \in E$.

A *path* is graph $P_n = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ such that $E = \{v_1v_2, v_2v_3, \ldots, v_{n-1}v_n\}$.

A *4-neighbor grid* is a graph given by the cartesian product of two paths. Other grid graphs will be introduced in Section 4.1.

A graph $G = (V, E)$ is *connected* if for each pair $u, v \in V$ there is a sequence of vertices $u = v_1, \ldots, v_k = v$ such that $v_i, v_{i+1} \in E$ for $1 \leq i \leq k-1$. If k is the smallest possible value for such sequence, $v_1, \ldots, v_k$ is a *shortest path* between u and v, and we note $\mathrm{dist}(u, v) = k$.

A *tree* $T = (V, E)$ is a connected graph such that C_i is not a subgraph of T for any $3 \leq i \leq n$. A *rooted tree* T is a tree with a distinguished vertex or node r called *root*. If $(a, b) \in E(T)$ we say that a is the *parent* of b and b the *child* of a when $\mathrm{dist}(r, a) < \mathrm{dist}(r, b)$. We call a *leaf* to a node without child nodes, and *internal node* to any node that has at least one child.

Given a graph $G = (V, E)$, an *independent set* of G is a set $I \subseteq V$ such that no two vertices in I are neighbors.

A *bipartite graph* is a graph whose set of vertices may be partitioned into two independent sets.

Let $n, k \in \mathbb{N}_0$, we denote $K_{n,k} = (V_n \cup V_k, E)$ the graph with independent sets V_n, V_k of size n and k respectively, such that $vw \in E$ if and only if $v \in V_n$ and $w \in V_k$ or viceversa. We call $K_{n,k}$ an (n, k)-*complete bipartite* graph.

For a graph G we define the *complement graph* $\overline{G} := (V, \overline{E})$ where $\overline{E} := \{uv : uv \notin E \text{ and } u \neq v\}$. If $\mathcal{G}$ is a graph class, we define co-$\mathcal{G}$ the class of graphs whose complement is in $\mathcal{G}$.

Given a graph family $\mathcal{F}$, we say that G is $\mathcal{F}$-*free* if G has no induced subgraph in $\mathcal{F}$. Given a graph G and $F \in \mathcal{F}$ for a fixed finite family $\mathcal{F}$ we can take all the induced subgraphs of G with $|V(F)|$ vertices, this is $\binom{|V(G)|}{|V(F)|} \in \mathcal{O}(|V(G)|^{|V(F)|})$, since F is fixed checking if the subgraph is F is polynomial. Therefore we have that for each finite family $\mathcal{F}$ we can decide if a graph G is $\mathcal{F}$-free in polynomial time.

A *chordal graph* is a $\{C_i\}$-free graph for $i \geq 4$. A *split graph* $G = (V, E)$ is a graph such that there is a partition of V_G into K_n and I, where I is an independent set. Alternatively, G is a split graph if and only if G and $\overline{G}$ are chordal [7].

A *cograph* is a graph that can be constructed using the following rules

- A graph with a single vertex is a cograph.

- If G, H are cographs with disjoint vertex set then $G \cup H$ is a cograph.
- If G is a cograph then $\overline{G}$ is a cograph.

Another recursive definition is

- $(\{v\}, \emptyset)$ is a cograph.
- If G and H are cographs then $G \oplus H$ is a cograph.
- If G and H are cographs then $G \cup H$ is a cograph.

Alternatively, a graph G is a cograph if and only if G is P_4-free [6].

Given a set family $\mathcal{F} = \{F_1, \ldots, F_n\}$ the *intersection graph* $I(\mathcal{F}) := (\mathcal{F}, E)$ is such that $F_iF_j \in E$ if and only if $F_i \cap F_j \neq \emptyset$. A *interval graph* is the intersection graph of $\{I_1, \ldots, I_n\}$ where $I_i = [a_i, b_i] \subset \mathbb{R}$. A subclass of interval graphs will be studied in Chapter 4.

1.2.3. Complexity theory. In this section we give a short introduction to algorithmic complexity theory, we suggest [8] for a more comprehensive reading. First we give some formal definitions assuming the reader is familiar with basic notions of what a Turing Machine (TM) and Non Deterministic Turing Machine (NDTM) are, then we give some examples and conclude this section with an introduction to GRAPH-ISOMORPHISM problem and its complexity.

Given Σ a finite set of symbols which we call *alphabet*, we say that a finite ordered sequence of elements or characters of Σ is a *string*. We note the set of all strings of Σ as Σ^*. A *language* over Σ is a set $\mathcal{L} \subseteq \Sigma^*$. If there exists a TM that accepts $\mathcal{L}$ we say that $\mathcal{L}$ is decidable or computable. If $\mathcal{L}$ is computable such that there is a TM $\mathcal{M}$ and a polynomial P such that $\mathcal{M}$ decides if $s \in \mathcal{L}$ in $T(s)$ steps with $T(s) \leq P(|s|)$ for each $s \in \Sigma^*$ then we say that $\mathcal{M}$ is polynomial, or that there is a polynomial algorithm for $\mathcal{L}$. If there is a NDTM that computes $\mathcal{L}$ in a time bounded by a polynomial in the size of the input, we say that $\mathcal{L}$ is $\mathcal{NP}$. Given the languages $\mathcal{L}$ and $\mathcal{L}'$, if there is a mapping $f : \mathcal{L} \longrightarrow \mathcal{L}'$ computable in polynomial time, and $s \in \mathcal{L}$ if and only if $f(s) \in \mathcal{L}'$, then we note $\mathcal{L} \leq_P \mathcal{L}'$ and we say that $\mathcal{L}$ is polynomially reducible to $\mathcal{L}'$. Given $\mathcal{L}'$, if for each $\mathcal{L} \in \mathcal{NP}$ we have $\mathcal{L} \leq_P \mathcal{L}'$ then we say that $\mathcal{L}'$ is $\mathcal{NP}$-hard. The set of languages $\mathcal{NP}$-complete is defined as the intersection of $\mathcal{NP}$-hard and $\mathcal{NP}$.

An alternative and more used definition is based on decision problems. A *decision problem* Π is to answer YES or NO for each valid input or instance for Π, if there is an algorithm that finishes in a polynomial number of steps on the size of the input then we say that Π is a polynomial problem or *tractable*. If Π admits a *positive certificate* for each YES instance, that is, some extra information that can be used to check if the answer is indeed YES in polynomial time, then we say that Π is $\mathcal{NP}$. If the same holds for *negative certificates* and NO instances, then we say that Π is co-$\mathcal{NP}$. Given the problems Π' and Π, if we can transform in polynomial time any instance I of Π into some instance I' of Π' such that I is a YES instance if and only if I' is a YES instance, then we say that Π is reducible to Π', and we note $\Pi \leq_P \Pi'$. If $\Pi \leq_P \Pi'$ for each $\Pi \in \mathcal{NP}$ we say that Π' is $\mathcal{NP}$-hard. Finally the set of $\mathcal{NP}$-complete problems are those in $\mathcal{NP}$ and in $\mathcal{NP}$-hard.

For example, any polynomial problem is $\mathcal{NP}$, because we may use the same input as a positive certificate, and run the polynomial algorithm to check if the answer is indeed YES. On the other hand, if we can solve in polynomial time some $\mathcal{NP}$-hard problem then we are able to solve in polynomial time any $\mathcal{NP}$ problem, of course solving polynomially some $\mathcal{NP}$-hard problem it is not an easy task. Nowadays in the scientific community there is a wide belief that there is no polynomial time algorithm for solving some $\mathcal{NP}$-complete problem. Actually the question of whether $\mathcal{P} = \mathcal{NP}$ or $\mathcal{P} \neq \mathcal{NP}$ is one of the most important open problems in complexity theory up to day. In 1971 a breakthrough was made by Stephen A. Cook when he proved in [5] that the boolean satisfiability problem is $\mathcal{NP}$-complete. In USSR a similar work was published in 1973 by Leonid A. Levin, where six different $\mathcal{NP}$-complete problems were given [11]. After this result many problems were proved to be $\mathcal{NP}$-complete using polynomial reductions. Knowing that some problem is $\mathcal{NP}$-hard usually is a good flag to stop searching for exact polynomial algorithms, and suggests to explore other approaches, like approximation algorithms, integer programming techniques, heuristics or exponential algorithms with low exponents or good average running time. The complexity of many problems was extensively studied for different restrictions, yielding some useful classification in polynomial cases and $\mathcal{NP}$-hard cases. The goal of this work is to study the complexity of one particular problem and classify restrictions of this problem into polynomial and $\mathcal{NP}$-hard.

For example, given a boolean formula the BOOLEAN-SATISFIABILITY problem is to decide if there is an assignment to its variables such that the formula is true under such assignment, this problem is $\mathcal{NP}$-complete. Another well known $\mathcal{NP}$-complete example is the MAXIMUM-CLIQUE problem, which consists of finding the largest complete subgraph in the input graph, its decision version is to decide if there is a complete subgraph of size at least k in G. Another well known $\mathcal{NP}$-complete problem is the TRAVELLING-SALESMAN problem, which given a complete graph with an edge weight function $w : E \longrightarrow \mathbb{Q}_{\geq 0}$, is to decide if there is a cycle that contains all the vertices and the sum of all edges on the cycle is less or equal than some value. The optimization version of this problem is to find the cycle with the smallest edge sum possible. As we stated before, to prove that some problem is $\mathcal{NP}$-complete we can make a polynomial reduction from another $\mathcal{NP}$-complete problem, we will do this many times in this work, introducing more $\mathcal{NP}$-complete problems. Many $\mathcal{NP}$-complete problems are not related directly to graph theory, for the interested reader we suggest the appendix of [8].

Two graphs, $G = (V, E)$ and $H = (V', E')$, are *isomorphic* if there is a 1-1 mapping $f : V \longrightarrow V'$ such that $uv \in E$ if and only if $f(u)f(v) \in E'$. The GRAPH-ISOMORPHISM problem is to decide if two graphs are isomorphic. Clearly a 1-1 mapping may be used as a positive certificate, hence this problem is $\mathcal{NP}$. In [8] the authors mentioned this problem and stated that its complexity is yet unkown. Nowadays, 3 decades later, GRAPH-ISOMORPHISM has its own complexity class because no one was able to give a polynomial algorithm or show that it is $\mathcal{NP}$-complete. Although in practice this problem admits very fast algorithms and many techniques were introduced since 1979, for example, in [13] and [15] are presented exact algorithms that in the worst case have exponential running time, but in practical applications they perform well. The most studied approaches are based on computing some information on both graphs and compare it, this usually provides a fast mechanism to state that two graphs are not isomorphic, but finding an exhaustive list of what information to compute to get an exact and fast algorithm is a major task. Another widespread approach is to compare the graphs directly.

The class $\mathcal{GI}$ is the set of problems that can be solved with a polynomial reduction to GRAPH-ISOMORPHISM, a problem is $\mathcal{GI}$-hard if there is a polynomial reduction from any $\mathcal{GI}$ problem to this problem, and a problem is $\mathcal{GI}$-complete if it is $\mathcal{GI}$ and $\mathcal{GI}$-hard. In this work we will use only polynomial-time Turing reductions, which were described above. For a more comprehensive reading on $\mathcal{GI}$ and related classes, based on other reductions and hierarchies, we suggest [**16**] and [**10**].

In [**16**] the author states that GRAPH-ISOMORPHISM is not likely to be $\mathcal{NP}$-complete, based on the observation that the counting version of GRAPH-ISOMORPHISM, which counts the number of different isomorphisms is as difficult as the decision problem. This is not the case with $\mathcal{NP}$-complete problems and the related counting class $\#\mathcal{P}$, where YES solutions are counted. He mentions that most counting versions of $\mathcal{NP}$-complete problems are $\#\mathcal{P}-$*complete*, and at the time of writing it is not even known if $\#\mathcal{P}$ is included in the polynomial hierarchy. Since the results presented in [**16**], more related work was done and the belief that $\mathcal{GI}$ is not in $\mathcal{NP}$-complete became wide spreaded, a summary of possible arguments may be found in [**2**].

In this work we will show some of the relations between one particular problem and the $\mathcal{GI}$ class.

1.2.4. Maximum common edge subgraph problem.

Definition 1.2.1. *Given two graphs $G = (V_G, E_G)$, $H = (V_H, E_H)$ with $n_G = n_H$ and a 1-1 mapping $f : V_G \longrightarrow V_H$. We define γ_f to be the* cardinality *of f, given by $\gamma_f := |\{uv \in E_G : f(u)f(v) \in E_H\}|$. If obvious, we ommit the f subindex. We denote $\Gamma(G,H) = \max_f\{\gamma_f\}$ and $\Psi(G,H) = \min_f\{\gamma_f\}$.*

Definition 1.2.2. Maximum common edge subgraph problem (MCESP). *Given two undirected graphs G and H with the same number of vertices, the* MCESP *asks for a 1-1 mapping $f : V(G) \longrightarrow V(H)$ such that $\gamma_f = \Gamma(G,H)$. The decision version of* MCESP *is, given an instance (G,H,k) with $|V(G)| = |V(H)|$, decide if $\Gamma(G,H) \geq k$.*

Naturally a similar problem arises, instead of asking for the 1-1 mapping of the largest cardinality we may ask for the smallest cardinality.

Definition 1.2.3. Minimum common edge subgraph problem (MinCESP). *Given two undirected graphs G and H with the same number of vertices, the* MINCESP *asks for a 1-1 mapping $f : V(G) \longrightarrow V(H)$ such that $\gamma_f = \Psi(G,H)$. The decision version of* MINCESP *is, given an instance (G,H,k) with $|V(G)| = |V(H)|$, decide if $\Psi(G,H) \leq k$.*

We will show several relations between MINCESP and MCESP in Section 2.1, these will be useful to relate complexity results of both problems.

Observation 1.2.1. MCESP $\in \mathcal{NP}$ *and* MinCESP $\in \mathcal{NP}$.

Proof. Given a 1-1 map $f : V(G) \longrightarrow V(H)$, γ_f can be computed in $\mathcal{O}(m)$. ∎

If $\mathcal{G}$ and $\mathcal{H}$ are graph classes, we will denote MCESP($\mathcal{G}, \mathcal{H}$) to the MCESP problem with one input graph restricted to $\mathcal{G}$ and the other to $\mathcal{H}$, same notation will be used for MinCESP.

Given G a graph, the Hamiltonian-Circuit problem consists in deciding if there is a *Hamiltonian circuit* in G, that is, a circuit that contains all the vertices of G exactly once. This is equivalent to say that C_n is a subgraph of G. The Hamiltonian-Path problem consists in deciding if there is a *Hamiltonian path* in G, that is, a path that contains all the vertices of G exactly once. This is equivalent to say that P_n is a subgraph of G. The Maximum-Clique problem consists in finding the largest complete subgraph of G, its decision version is, given $k \in \mathbb{N}$, decide if K_k is a subgraph of G. In [8] these three problems are shown to be $\mathcal{NP}$-complete.

Proposition 1.2.1. *The* MCESP *is* $\mathcal{NP}$*-complete.*

In the following we give three alternative proofs.

Proof 1. Deciding if a graph G admits a Hamiltonian circuit is deciding if C_n is a subgraph of G, this gives a trivial reduction from Hamiltonian-Circuit. Given a graph G with n vertices, if $\Gamma(G, C_n) \geq n$ then all the edges of C_n contributed to the solution. Since the solution has an associated 1-1 mapping $f : V(C_n) \longrightarrow V(G)$, if C_n is the circuit $v_1 \ldots, v_n$ then $f(v_i)f(v_{i+1})$ for $1 \leq i \leq n-1$ and $f(v_n)f(v_1)$ are adjacent in G. This forms a circuit in G of length n, therefore G admits a Hamiltonian Circuit. Reciprocally, if there is a Hamiltonian Circuit $v_1, \ldots, v_n$ in G, and the cycle of C_n is $w_1, \ldots, w_n$, we define $f(w_i) = v_i$ for $1 \leq i \leq n$, clearly all the edges of C_n are contributing to γ_f, then $\Gamma(G, C_n) \geq \gamma_f = n$. ∎

Proof 2. The second proof consists in reducing from Hamiltonian-Path, the arguments are ommited since the proof is essentially the same. ∎

Proof 3. Deciding if a graph G with n vertices contains a complete subgraph of k vertices is the same as deciding if

$$\Gamma(G, K_k \cup \overline{K_{n-k}}) \geq k(k-1)/2,$$

this leads to a trivial reduction from Maximum-Clique. Given an instance (G, k) of Maximum-Clique, if $\Gamma(G, K_k \cup \overline{K_{n-k}}) \geq k(k-1)/2$ then all the edges of K_k are contributing to the solution, take the associated 1-1 mapping $f : V(K_k \cup \overline{K_{n-k}}) \longrightarrow V(G)$, denote

$$V(K_k \cup \overline{K_{n-k}}) = \{v_1, \ldots, v_k, v_{k+1}, \ldots, v_n\}$$

where v_iv_j are adjacent if and only if $i, j \leq k$, then the induced subgraph $G[\{f(v_1), \ldots, f(v_k)\}]$ must be complete, otherwise some edge of K_k wouldn't contribute. This implies that G contains a complete graph of k vertices. Reciprocally, if G has a clique with k or more vertices, then this clique contains a complete subgraph of k vertices, we can map injectively the vertices of K_k on this complete subgraph and extend it to a 1-1 mapping arbitrarily, leading to $\gamma_f = k(k-1)/2 \leq \Gamma(G, K_k \cup \overline{K_{n-k}})$. ∎

Note that these three proofs give us the following

Corollary 1.2.1. *If $\mathcal{G}$ is a class of graphs where* Hamiltonian-Circuit, Hamiltonian-Path *or* Maximum-Clique *is $\mathcal{NP}$-complete, then* MCESP *is $\mathcal{NP}$-complete if we restrict one of the input graphs to $\mathcal{G}$.*

Furthermore, the idea of the proof which uses a reduction from Maximum-Clique leads us to the following

Theorem 1.2.1. *If $\mathcal{H}$ is a class of graphs closed for addition of isolated vertices, i.e. if $(V, E) \in \mathcal{H}$ then $(V \cup \{v\}, E) \in \mathcal{H}$ for $v \notin V$, and $\mathcal{G}$ is a class of graphs such that deciding if $H \in \mathcal{H}$ is a subgraph of $G \in \mathcal{G}$ is $\mathcal{NP}$-complete, then* MCESP$(\mathcal{G}, \mathcal{H})$ *is $\mathcal{NP}$-complete.*

Proof. Let $H \in \mathcal{H}$ and $G \in \mathcal{G}$, we are to decide if H is a subgraph of G. If $n_H > n_G$ the answer is clearly No, therefore suppose $n_H \leq n_G$. Add $n_G - n_H$ isolated vertices to H, name the resulting the graph H'. By hypothesis $H' \in \mathcal{H}$ and by construction $n_{H'} = n_G$. If $\Gamma(H', G) \geq m_H$ then $\Gamma(H', G) = m_H$, because in H' there are only m_H edges. Hence H' is a subgraph of G and by construction H is a subgraph of H', therefore H is a subgraph of G. Reciprocally if H is a subgraph of G, then H' is also a subgraph of G, yielding $\Gamma(H', G) = m_{H'} = m_H$. ∎

The ease of the reductions used in this first results leads us to think that MCESP is a very difficult problem compared to other $\mathcal{NP}$-complete problems.

1.3. About this work

In this chapter we gave the basic notions for understanding this work, introducing the required definitions and basic results. There is no precedence order between Chapter 2, Chapter 3 and Chapter 4, so the reading may be in any order.

In Chapter 2 we explore general results related to MCESP and the behavior of 1-1 mappings. The main result of Section 2.1 is that MCESP$(\mathcal{G}, \mathcal{H})$ is $\mathcal{NP}$-complete if and only if MCESP(co-$\mathcal{G}$, co-$\mathcal{H}$) is $\mathcal{NP}$-complete, to achieve this result we relate the complexities of MinCESP and MCESP. In Section 2.2 we explore the behavior of 1-1 mappings when both graphs are restricted to have same number of edges and vertices, within this class of graphs we define a graph distance function related to MCESP. In Section 2.3 we formalize the fact that MCESP is a generalization of Graph-Isomorphism and relate the $\mathcal{GI}$ complexity with MCESP.

In Chapter 3 we study the MCESP problem when one of the graphs is a complete bipartite graph. We explore the problem complexity when the second graph is unrestricted, complete bipartite, cograph, bipartite and union of stars. We prove that the first case is $\mathcal{NP}$-complete and we give some observations on the bipartite case, which are partial results that may be used to prove that the restriction is $\mathcal{NP}$-complete or give a polynomial time algorithm. On the positive side, we give polynomial time algorithms for the rest of the restrictions.

In Chapter 4 we explore more restrictions to both graphs. In Section 4.1 we show that MCESP is $\mathcal{NP}$-complete when one graph is a grid and the other is a union of grids, this holds for grids with 4, 6 and 8 neighbors. We extend this idea to honeycomb grids. In the remaining part of Chapter 4 we explore the cases when both graphs are restricted to split graphs, conntected proper interval graphs, trees and unions of paths. We also explore the case when the first graph is bipartite and the second a complete bipartite with isolated vertices. For all of these restrictions we prove that MCESP is $\mathcal{NP}$-complete.

CHAPTER 2

Mapping Structure

In this chapter we explore general results related to MCESP and the behavior of 1-1 mappings. In Section 2.1 we show a relation between $\Gamma(G,\overline{H})$ and $\Psi(G,H)$ which enables us relate the complexities of MinCESP and MCESP, based on this result we prove that $\Gamma(\overline{G},H) = \Gamma(G,\overline{H}) + m_H - m_G$. Using these relations we observe that MCESP$(\mathcal{G},\mathcal{H})$ is $\mathcal{NP}$-complete if and only if MCESP(co-$\mathcal{G}$, co-$\mathcal{H}$) is $\mathcal{NP}$-complete. In Section 2.2 we explore the behavior of 1-1 mappings when both graphs are restricted to have same number of edges and vertices, we prove some technical results that enable us to show a distance function related to MCESP. Finally in Section 2.3 we formalize the fact that MCESP is a generalization of Graph-Isomorphism and relate the $\mathcal{GI}$ complexity with MCESP.

2.1. General results on Γ

Theorem 2.1.1. *If G and H are two graphs with $n_G = n_H$, then $\Gamma(G,\overline{H}) = m_G - \Psi(G,H)$.*

Proof. Let $\Gamma(G,H) = t$, then there is a 1-1 mapping $f : V_G \longrightarrow V_H$ such that its cardinality is t. Let T be the graph whose edge set are the edges that are in G and in H via f, define $E_1 := E_G \setminus E_T$, $E_2 := E_H \setminus E_T$. Since the edges of T are those in G and H we have that E_T, E_1 and E_2 are pairwise disjoint sets. Consider $\overline{H}$, the edges from E_1 are clearly contained in $\overline{H}$ under the mapping f, and the edges of T and E_2 are not in $\overline{H}$ under f. Using the mapping f for $(G,\overline{H})$ we have $\Psi(G,\overline{H}) \leq \gamma_f(G,\overline{H}) = |E_1| = |E_G \setminus E_T| = m_G - \Gamma(G,H)$.

Let $g : V_G \longrightarrow V_{\overline{H}}$ the 1-1 mapping such that $\gamma_g = \Psi(G,\overline{H})$. Let R be the graph whose edge set are the edges that are in G and in $\overline{H}$ via g, define $E_1 := E_G \setminus E_R$, $E_2 := E_{\overline{H}} \setminus E_R$. Since the edges of R are those in G and $\overline{H}$ we have that E_R, E_1 and E_2 are pairwise disjoint sets. Consider $\overline{\overline{H}} = H$, the edges from E_1 are clearly contained in H under the mapping g, and the edges of R and E_2 are not in H under g. Using the mapping g for (G,H) we have

$$\Gamma(G,H) \geq \gamma_g(G,H) = |E_1| = |E_G \setminus E_R| = m_G - \Psi(G,\overline{H}).$$

Therefore $\Psi(G,\overline{H}) \geq m_G - \Gamma(G,H)$. ■

Corollary 2.1.1. *Given $\mathcal{G}$ and $\mathcal{H}$ graph classes, then* MCESP$(\mathcal{G},\mathcal{H})$ *is $\mathcal{NP}$-complete if and only if* MinCESP$(\mathcal{G}$, *co-*$\mathcal{H})$ *is $\mathcal{NP}$-complete.*

PROOF. First we prove MCESP($\mathcal{G},\mathcal{H}$) $\leq_P$ MINCESP($\mathcal{G}$, co-$\mathcal{H}$). Let (k, G, H) be an instance of MCESP($\mathcal{G},\mathcal{H}$) where $G \in \mathcal{G}$ and $H \in \mathcal{H}$, construct the instance $(m_G - k, G, \overline{H})$ for MINCESP($\mathcal{G}$, co-$\mathcal{H}$), then using Theorem 2.1.1

$$\Gamma(G,H) \geq k \iff m_G - \Psi(G,\overline{H}) \geq k \iff m_G - k \geq \Psi(G,\overline{H}).$$

In the following we prove that MINCESP($\mathcal{G}$, co-$\mathcal{H}$) $\leq_P$ MCESP($\mathcal{G},\mathcal{H}$). Let $(k, G, \overline{H})$ be an instance of MINCESP($\mathcal{G}$, co-$\mathcal{H}$) where $G \in \mathcal{G}$ and $\overline{H} \in$ co-$\mathcal{H}$, construct the instance $(m_G - k, G, H)$ for MCESP($\mathcal{G},\mathcal{H}$), then

$$\Psi(G,\overline{H}) \leq k \iff m_G - \Gamma(G,H) \leq k \iff m_G - k \leq \Gamma(G,H).$$

■

Definition 2.1.1. *We call $\mathcal{G}_{n,m}$ the family of graphs with n vertices and m edges.*

Corollary 2.1.2. *If G and H are graphs with $n_G = n_H$, then $\Gamma(\overline{G},H) = \Gamma(G,\overline{H}) + m_H - m_G$, and $\Psi(\overline{G},H) = \Psi(G,\overline{H}) + m_H - m_G$. Observe that if $G, H \in \mathcal{G}_{n,m}$ then $\Gamma(G,\overline{H}) = \Gamma(\overline{G},H)$, and $\Psi(G,\overline{H}) = \Psi(\overline{G},H)$.*

PROOF. Observe using Theorem 2.1.1 that

$$\Gamma(G,H) = m_G - \Psi(G,\overline{H})$$

$$\Gamma(G,H) = m_H - \Psi(\overline{G},H)$$

therefore we get $\Psi(\overline{G},H) = \Psi(G,\overline{H}) + m_H - m_G$. In the same way we prove $\Gamma(\overline{G},H) = \Gamma(G,\overline{H}) + m_H - m_G$. ■

Observation 2.1.1. *Given $\mathcal{G}$ and $\mathcal{H}$ graph classes, then* MCESP(*co-*$\mathcal{G},\mathcal{H}$) *is $\mathcal{NP}$-complete if and only if* MCESP($\mathcal{G}$, *co-*$\mathcal{H}$) *is $\mathcal{NP}$-complete. Also note that* MCESP($\mathcal{G},\mathcal{H}$) *is $\mathcal{NP}$-complete if and only if* MCESP(*co-*$\mathcal{G}$, *co-*$\mathcal{H}$) *is $\mathcal{NP}$-complete. Furthermore, this relation holds for the polynomial case, in the sense that* MCESP($\mathcal{G},\mathcal{H}$) *admits a polynomial time algorithm if and only if* MCESP(*co-*$\mathcal{G}$, *co-*$\mathcal{H}$) *admits a polynomial time algorithm.*

2.2. A distance between graphs using MCESP

Definition 2.2.1. *Let $G, H \in \mathcal{G}_{n,m}$, we define* $\mathrm{dist}(G,H) := m - \Gamma(G,H)$.

Lemma 2.2.1. *If $G, H, H' \in \mathcal{G}_{n,m}$ with $|E_H \triangle E_{H'}| = 2$, then $|\Gamma(G,H) - \Gamma(G,H')| \leq 1$.*

PROOF. Assume w.l.o.g. $\Gamma(G,H) \geq \Gamma(G,H')$ and $|\Gamma(G,H) - \Gamma(G,H')| > 1$, then we have $\Gamma(G,H) > 1 + \Gamma(G,H')$. Let $f : V_G \longrightarrow V_H$ be the 1-1 mapping such that $\gamma_f = \Gamma(G,H)$. Since $|E_H \triangle E_{H'}| = 2$ we lose at most one common-edge by using f as mapping for (G,H') instead of (G,H), hence $\gamma_f(G,H') \geq \gamma_f(G,H) - 1$, then

$$1 + \Gamma(G,H') \geq 1 + \gamma_f(G,H') \geq \gamma_f(G,H) = \Gamma(G,H),$$

a contradiction.

■

Let $G, H \in \mathcal{G}_{n,m}$ such that $\Gamma(G,H) < m$, let $f : V_G \longrightarrow V_H$ be an optimal 1-1 mapping. Consider the operation of deleting an edge $uv \in E_H$ such that $f^{-1}(u)f^{-1}(v) \notin E_G$ from E_H and adding an edge $f(x)f(y) \notin E_H$ such that $xy \in E_G$ to the edges of H, we denote H' to the obtained graph. By definition we get $\Gamma(G,H)+1 = \gamma_f(G,H')$, furthermore, since $|E_H \triangle E_{H'}| = 2$ we conclude by Lemma 2.2.1 that $\Gamma(G,H)+1 = \Gamma(G,H') = \gamma_f(G,H')$. Repeating this argument we can transform H into an isomorphic graph of G, if we only use the operations defined before, this algorithm takes $\text{dist}(G,H)$ operations.

Proposition 2.2.1. *The function* $\text{dist} : \mathcal{G}_{n,m}^2 \longrightarrow \mathbb{N}$ *is a distance over* $\mathcal{G}_{n,m}$ *considering the graph isomorphism as equality.*

Proof. Let $G, H \in \mathcal{G}_{n,m}$, recall from Definition 1.2.1 that $\Gamma(G,H) = |\{uv \in E_G : f(u)f(v) \in E_H\}| \leq m$, then $\text{dist}(G,H) \geq 0$.

We have $\text{dist}(G,H) = 0 \iff m - \Gamma(G,H) = 0 \iff$ there is a 1-1 mapping $f : V_G \longrightarrow V_H$ such that $|\{uv \in E_G : f(u)f(v) \in E_H\}| = m \iff [uv \in E_G \iff f(u)f(v) \in E_H]$, using that both graphs are in $\mathcal{G}_{n,m}$ we get that G is isomorphic to H.

Let $G, H, F \in \mathcal{G}_{n,m}$, using the above procedure we can transform G into F in $\text{dist}(G,F)$ steps and then F into H in $\text{dist}(F,H)$ steps, on the other hand we can transform G into H in $\text{dist}(G,H)$ steps. This procedure improves the distance by exactly one unit at each step, then $\text{dist}(G,H) \leq \text{dist}(G,F) + \text{dist}(F,H)$.

The symmetry holds trivially. ■

2.3. Graph Isomorphism

The next proposition shows that MCESP is a generalization of Graph-Isomorphism.

Proposition 2.3.1. *If* Graph-Isomorphism *is* $\mathcal{GI}$*-complete when restricted to the class of graphs* $\mathcal{H}$*, then* MCESP *is* $\mathcal{GI}$*-hard when both graphs are restricted to* $\mathcal{H}$*.*

Proof. Given $G, H \in \mathcal{H}$, then G is isomorphic to H if and only if there is a 1-1 mapping $f : V_G \longrightarrow V_H$ such that $uv \in E_G$ if and only if $f(u)f(v) \in E_H$. Since f is a 1-1 mapping we have $n_G = n_H$, and by the edge conservation property of f we have that $m_G = m_H$. Thus, deciding if there is an isomorphism is equivalent to decide if $n_G = n_H$, $\Gamma(G,H) = m_G$ and $m_G = m_H$. ■

Observation 2.3.1. *There are classes of graphs such that if both input graphs are restricted to such class, the* MCESP *is* $\mathcal{NP}$*-complete, but the complexity of* Graph-Isomorphism *with the same restriction is unkown, for instance, this happens with bipartite graphs. This is similar to what happens with* Subgraph-Isomorphism *problem and* Graph-Isomorphism*.*

CHAPTER 3

Complexity over Complete Bipartite graphs

In this chapter we study the MCESP problem when one of the graphs is a complete bipartite graph. We explore the problem complexity when the second graph is unrestricted, complete bipartite, cograph, bipartite and union of stars. We prove that the first case is $\mathcal{NP}$-complete and we give some observations on the bipartite case, these observations are partial results that may be used in further research of the complexity of this restriction. On the positive side, we give polynomial time algorithms for the rest of the restrictions.

3.1. Complete Bipartite vs. arbitrary graph

Given a graph G, the problem of finding a set of vertices $S \subseteq V_G$ such that the number of edges between S and $V_G \setminus S$ is maximized, is called the Max-Cut problem. In [8] Max-Cut is shown to be $\mathcal{NP}$-complete. The Max-Cut problem can be restated as to find a maximum edge bipartite subgraph of G. The decision version of this problem consists in deciding if there is a *cut* (or bipartite subgraph) of edge size greater or equal to k.

Proposition 3.1.1. *The* MCESP(G, H) *with the restriction* $H = K_{n,k}$ *is* $\mathcal{NP}$*-complete.*

Proof. Recall from Observation 1.2.1 that MCESP $\in \mathcal{NP}$. In the following we prove that Max-Cut $\leq_P$ MCESP. Let (G, k) be an instance of Max-Cut decision problem, where $k \in \mathbb{N}$ and G is an arbitrary graph. Consider the instances $I_0, \ldots, I_{|V|} = (K_{0,|V|}, G), (K_{1,|V|-1}, G), \ldots, (K_{|V|,0}, G)$ for MCESP, we note $\Gamma_i =$ MCESP(I_i), take the instance I_i that maximizes Γ_i over all the instances. By definition Γ_i is the edge count of the maximum common subgraph of $K_{i,|V|-i}$ and G. Therefore the answer for Max-Cut is Yes if $\Gamma_i \geq k$ and No otherwise. For completion we must remark that if B is a bipartite subgraph of G and $\{V_1, V_2\}$ any bipartition of V_B, we can add $\{v_1, \ldots, v_t\}$ isolated vertices to V_1 until we get $t + |V_1| + |V_2| = |V_G|$, this is a subgraph of $K_{t+|V_1|,|V_2|}$ and a subgraph of G; since the isolated vertices do not influence the edge count, it is enough to consider those instances for MCESP. ∎

In [3] the authors proved that for split, tripartite and co-bipartite Max-Cut is $\mathcal{NP}$-complete, a *tripartite* graph is one that admits a partition of its vertices into three independent sets. Using the same ideas of the proof of Proposition 3.1.1 we get the following

Corollary 3.1.1. MCESP$(K_{n,k}, G)$ *for* G *chordal, split, tripartite or co-bipartite graph is* $\mathcal{NP}$*-complete.*

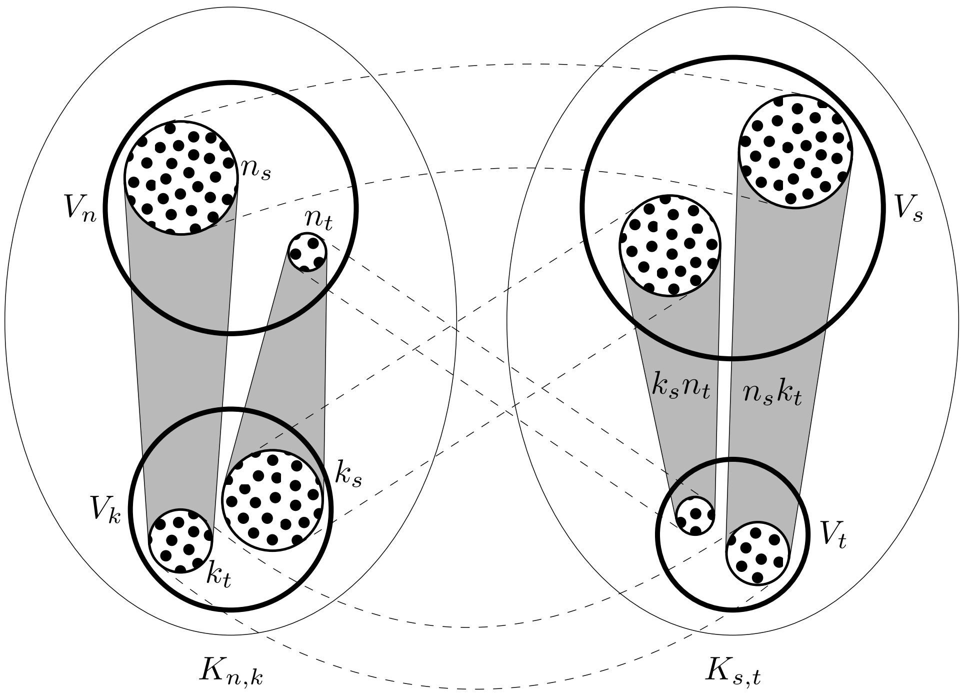

FIGURE 3.2.1. Arbitrary 1-1 mapping $g : V(K_{n,k}) \longrightarrow V(K_{s,t})$ and the edges contributing to γ_g.

3.2. Complete Bipartite vs. Complete Bipartite graph

Consider the graphs $K_{n,k}, K_{s,t}$ such that $k+n = s+t$, we can suppose without loss of generality that $n \leq k$, $t \leq s$, and $t \leq n$. Let us note $V_n = \{v_1, \ldots, v_n\}$, $V_k = \{v_{n+1}, \ldots, v_{n+k}\}$, $V_t = \{w_1, \ldots, w_t\}$, and $V_s = \{w_{t+1}, \ldots, w_{t+s}\}$.

Proposition 3.2.1. *Let $f : V_n \cup V_k \longrightarrow V_s \cup V_t$ such that $f(v_i) = w_i$, then $tk = \gamma_f = \Gamma(K_{n,k}, K_{s,t})$.*

PROOF. We have $t \leq n$ and $t + s = n + k$, then $s \geq k$. Note that for all $v \in V_n$ $\deg(v) = k$ and for all $w \in V_t$ $\deg(w) = s$. Therefore, since f maps t vertices from V_n to V_t and k vertices from V_k to V_s we get $tk = \gamma_f$.

We next prove that any 1-1 mapping g has cardinality less or equal to tk. Note a_b the number of vertices mapped from set V_a to V_b, where $a \in \{n, k\}$ and $b \in \{s, t\}$. The following holds:

$$\begin{aligned} t \leq n \leq k &\leq s \\ n_t &\leq t \\ k_t &\leq t \\ k_s &\leq k \\ n_t + k_t &= t. \end{aligned}$$

In Figure 3.2.1 the dotted lines represents the mapping of g, and the shaded area represents the edges that are part of γ_g. Therefore $\gamma_g = n_s k_t + k_s n_t =$

$\min\{n_s, k_s\}t + (k_s - \min\{n_s, k_s\})n_t + (n_s - \min\{n_s, k_s\})k_t$. Consider the following cases

- $n_s \leq k_s$: then $\gamma_g = n_s t + (k_s - n_s)n_t \leq n_s t + (k_s - n_s)t = k_s t \leq kt$.
- $n_s > k_s$: then $\gamma_g = k_s t + (n_s - k_s)k_t \leq k_s t + (n_s - k_s)t = n_s t \leq kt$.

■

3.3. Complete Bipartite vs. Cographs

We already saw a relation of Max-Cut and MCESP restricted to complete bipartite graphs, as we saw in the analyzed cases, when Max-Cut is $\mathcal{NP}$-complete so is MCESP restricted to complete bipartite graphs. In [3] it was shown that Max-Cut restricted to cographs is polynomial. In the following we present a polynomial dynamic programming algorithm for MCESP$(K_{n,k}, G)$ where G is a cograph. Recall from Subsection 1.2.2 that cographs may be defined in the following recursive manner

- $(\{v\}, \emptyset)$ is a cograph.
- If G and H are cographs then $G \oplus H$ is a cograph.
- If G and H are cographs then $G \cup H$ is a cograph.

Using this definition we may represent every cograph G with a *cotree*, which is a rooted tree whose leaves are nodes of G, and interior nodes represent unions or joins of the cographs represented by its child subtrees. We assume that the input is given as a binary cotree, for details on cographs we suggest the fundational work [6].

Intuitively our algorithm considers all the possible instances of MCESP for each subtree of the cotree that may fit in the input complete bipartite graph and have the correct number of vertices. We then define two recursive rules, for the join and the union of two cographs, each of these rules are polynomially computable given the optimal solution of the subinstances, which are calculated and stored in runtime memory. Before the algorithm and its analysis we introduce one well-known technical result, for the sake of completeness we give a proof.

Definition 3.3.1. *A* full binary tree *(FBT) is a binary rooted tree where each node is either a leaf or it is parent of exactly two nodes.*

Lemma 3.3.1. *(FBT Theorem) Any non-empty FBT with n internal nodes has exactly $n + 1$ leaves.*

Proof. If $n = 0$ we have one isolated vertex which is a leave. If $n = 1$ then we have two leaves. Let T be an FBT with $n + 1$ internal nodes, let x be an internal node with two child leaves, remove these two leaves, this yields an FBT with n internal nodes because the only node that is no longer internal is x. By inductive hypothesis this tree has $n + 1$ leaves, if we add back the removed leaves, x becomes an internal node and the leaf count is increased by 1, therefore we get $n + 2$ leaves for T. ■

Proposition 3.3.1. *There is a polynomial time algorithm for* MCESP *when restricted to complete bipartite graphs and cographs.*

Proof. Let G a cograph represented by a cotree T where T is a FBT, and $K_{n,k}$ the input complete bipartite graph, we will make recursion on T.

The base case is given by $\Gamma((\{v\},\emptyset),K_{1,0})=0$. In the recursive step, suppose we are making an operation on H and J where both are cographs represented by subtrees of T, and we have the values $\Gamma(K_{s,t},H)$ and $\Gamma(K_{l,r},J)$ for each $s+t=n_H$ (a), $l+r=n_J$ (b), $l+s=n$ (c) and $t+r=k$ (d). We call V_k to the subset of k vertices of $K_{n,k}$ and V_n to the subset of n vertices, where V_n and V_k are independent sets and $\{V_n,V_k\}$ is a partition of the vertices of $K_{n,k}$. In both recursions $V_l\cup V_s$ will be mapped to V_n and $V_t\cup V_r$ to V_k.

In case of $G=H\cup J$ no new edges were added when making the union, this is illustrated in Figure 3.3.1, we define $f:V(K_{n,k})\longrightarrow V_G$ such that

$$\gamma_f=\max\{\Gamma(K_{s,t},H)+\Gamma(K_{l,r},J):\text{(a),(b),(c),(d)}\}.$$

To see that $\gamma_f=\Gamma$ suppose there is a 1-1 mapping $g:V_G\longrightarrow V(K_{n,k})$ such that $\gamma_g>\gamma_f$, since there are no edges between H and J we can analyze $g\big|_{V_H}$ and $g\big|_{V_J}$ independently. We have that

$$\gamma_{g\big|_{V_H}}\leq\Gamma(K_{s,t},H)$$

where $s=|\{v\in V_H:g(v)\in V_n\}|$ and $t=n_H-s$. In the same way

$$\gamma_{g\big|_{V_J}}\leq\Gamma(K_{l,r},J)$$

where $l=|\{v\in V_J:g(v)\in V_n\}|$ and $r=n_J-l$. Thus we have

$$\Gamma(K_{s,t},H)+\Gamma(K_{l,r},J)\geq\gamma_{g\big|_{V_H}}+\gamma_{g\big|_{V_J}}=\gamma_g>\gamma_f,$$

a contradiction because of the definition of f. It is easy to see that there are at most $(n_G+1)(n_J+1)$ possible combinations for s,t,l and r, since $n_G\leq n+k$ and $n_J\leq n+k$ we have at most $(n+k+1)^2$ possible combinations.

In case of $G=H\oplus J$, n_Hn_J edges between H and J were added when joining, this is represented in Figure 3.3.2. We define $f:V(K_{n,k})\longrightarrow V_G$ such that

$$\gamma_f(G,K_{n,k})=\max\{sr+tl+\Gamma(K_{s,t},H)+\Gamma(K_{l,r},J):\ \text{(a),(b),(c),(d)}\}.$$

We can see that $\gamma_f=\Gamma$ using the same idea as before, the only thing that changes is the need of considering the new $tl+sr$ edges that are added to γ_g, where g is the 1-1 mapping used for the contradiction proof. Again we have at most $(n+k+1)^2$ combinations for the complete bipartite subinstances.

Since T is FBT and Lemma 3.3.1, we apply these rules to exactly $n+k-1$ interior nodes. When we compute the optimal values for each child node we can save the results to avoid recomputing them, this may be done using a matrix of size $(n+k-1)\times(n+k+1)\times(n+k+1)$, where the first coordinate denote the interior node of T, and the last two coordinates are related to the possible complete bipartite graphs for that node. With this strategy each node requires $\mathcal{O}((n+k)^2)$ steps to take the maximum value, using this for $n+k-1$ interior nodes this algorithm turns out to be $\mathcal{O}((n+k)^3)$.

■

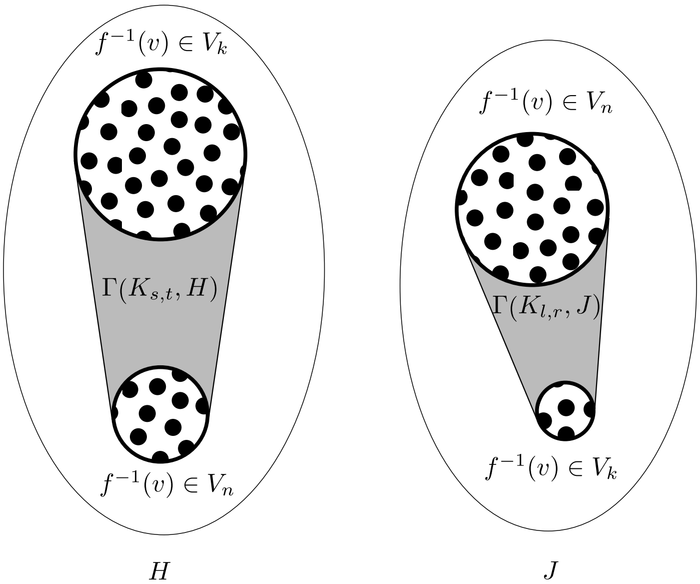

FIGURE 3.3.1. Recursive step for $H \cup J$. The 1-1 mapping $f : V(K_{n,k}) \longrightarrow V(H \cup J)$ where H and J are cographs, the shaded area represents edges that form part of γ_f.

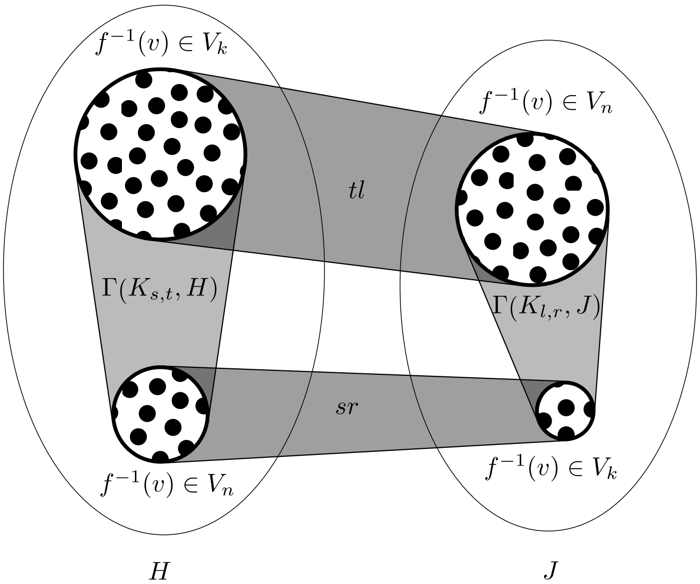

FIGURE 3.3.2. Recursive step for $H \oplus J$. The 1-1 mapping $f : V(K_{n,k}) \longrightarrow V(H \oplus J)$ where H and J are cographs, the shaded area represents edges that form part of γ_f.

3.4. Complete Bipartite vs. Bipartite graph

In this section we give partial results related to MCESP when one grpah is bipartite and the second a complete bipartite graph. We were not able to find a polynomial time algorithm neither give an $\mathcal{NP}$-completeness proof for this restriction. Nevertheless we think this restriction is $\mathcal{NP}$-complete due to the reformulation of the problem given in Lemma 3.4.1.

Let $K_{n,k}$ and G such that $|V(G)| = n + k$ and G a bipartite graph, let $t := \min\{n, k\}$. Since G is a bipartite graph we have

$$\mathrm{Adj}(G) = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}$$

with $B \in \{0,1\}^{|V_1|\times|V_2|}$ for some V_1 and V_2 independent sets of G that partition V_G. Note that $t \leq |V_1|$ or $t \leq |V_2|$, this can be proved trivially by contradiction. Consider selecting t rows or columns from B, and note this selection S. The value of the selection S, $v(S)$ is the sum of all the numbers in the selection, except those appearing in the position ij if the row i and column j are in S. We will note the set of rows in S with $\mathcal{R}$ and $\mathcal{C}$ to the set of columns in S.

Lemma 3.4.1. *Let $K_{n,k}$ and G such that $|V(G)| = n + k$ and G a bipartite graph, let $t := \min\{n, k\}$, then $\max_{S:|S|=t}\{v(S)\} = \Gamma(K_{n,k}, G)$.*

Proof. First, given a selection S we show how to construct an associated 1-1 mapping f, then we give an observation that enables us to get a selection for any 1-1 mapping, finally we show that $\gamma_f = v(S)$, where f is the associated mapping to S and vice versa.

Suppose S is a selection of size t for the matrix $B \in \{0,1\}^{|V_1|\times|V_2|}$, and assume the vertices of G indexed by the positions of $\mathrm{Adj}(G)$, that is, v_1 corresponds to the first row and the first column of $\mathrm{Adj}(G)$, v_2 to the second and so on. Suppose $n = t$, define $f : V_G \longrightarrow V(K_{k,n})$ as follows, for each selected row i let $f(v_i) \in V_n$, and for each selected column j let $f(v_{j+|V_1|}) \in V_n$, and map the remaining vertices to V_k such that f is a 1-1 mapping, this is well defined since $n = t$. A reverse construction gives a selection for every 1-1 mapping.

The edges $vw \in E_G$ that are contributing to γ_f are exactly those where $f(v) \in V_n$ and $f(w) \in V_k$. Therefore each vertex $v \in V_G$ such that $f(v) \in V_n$ is an endpoint of $\deg(v) - |\{w \in N(v) : f(w) \in V_n\}|$ contributing edges. Note that if an edge has an endpoint on v and is not contributing to γ_f, this will neccesarily have an endpoint on other vertex w such that $f(w) \in V_n$, then

$$\gamma_f = \sum_{v:f(v)\in V_n} \deg(v) - \frac{|\{w \in N(v) : f(w) \in V_n\}|}{2}.$$

On the other hand, $v(S)$ is the sum of all the selected entries in S, except the entires ij where the row i and column j are in S, thus

$$v(S) = \sum_{r\in\mathcal{R}} \deg(v_r) + \sum_{c\in\mathcal{C}} \deg(v_{c+|V_1|}) - \sum_{\substack{r\in\mathcal{R}\\ c\in\mathcal{C}}} B_{rc}.$$

By definition of f, the first two terms equals to $\sum_{v:f(v)\in V_n} \deg(v)$. If $r \in \mathcal{R}$, $B_{rc} = 1$ for some column c if and only if $v_r v_{c+|V_1|} \in E_G$, thus we can rewrite the last term as

$$\sum_{r\in\mathcal{R}} |\{w \in N(v_r) : w = v_{c+|V_1|} \text{ for some } c \in \mathcal{C}\}|.$$

By definition of f, $c \in \mathcal{C}$ if and only if $f(v_{c+|V_1|}) \in V_n$, then we rewrite this term as

$$\sum_{r\in\mathcal{R}} |\{w \in N(v_r) : f(w) \in V_n\}|.$$

In a similar way we can prove that

$$\sum_{\substack{r\in\mathcal{R}\\ c\in\mathcal{C}}} B_{rc} = \sum_{c\in\mathcal{C}} |\{w \in N(v_{c+|V_1|}) : f(w) \in V_n\}|.$$

Since V_1 and V_2 are independent sets, we get

$$2\sum_{r\in\mathcal{R}} |\{w \in N(v_r) : f(w) \in V_n\}| = \sum_{v:f(v)\in V_n} |\{w \in N(v) : f(w) \in V_n\}|,$$

hence $v(S) = \gamma_f$.

■

Observation 3.4.1. *Let $B \in \{0,1\}^{n\times(k+n+1)}$, and for $1 \le i \le n$ we have $\sum_{j=1}^{k+n+1} b_{ij} > n$, given an optimal selection $\mathcal{S}$ such that $|\mathcal{S}| = t \le n$, then $\mathcal{S} \subseteq \mathrm{Rows}(B)$.*

PROOF. Given $t \le n$, since each column can sum at most n, each row sums at least $n+1$, and we select at most n rows or columns, it is obvious that any row selection sums more than a selection that contains columns. Therefore the optimal solution is contained in the row set of B. ■

Observation 3.4.2. *If G, H are bipartite graphs such that $n_G = n_H = n$ and*

$$\Gamma(K_{i,n-i}, G) = \Gamma(K_{i,n-i}, H) \text{ for } 0 \le i \le n,$$

then not necessarily G and H are isomorphic. Consider

3.5. Complete Bipartite vs. union of Stars

A *star graph* S_i is a graph $K_{1,i}$ with $i \ge 0$, if $i = 0$ then the graph is an isolated vertex. In this section we present a polynomial dynamic programming algorithm for computing $\Gamma(K_{n,k}, G)$ where G is a union of stars. We think the complexity of this algorithm may be drastically improved, but we prefer a simpler approach with an easy proof for showing that this case is polynomial.

Proposition 3.5.1. *If $G = S_{n_1} \cup \cdots \cup S_{n_t}$ such that $t + \sum_{i=1}^{t} n_i = n + k$ then $\mathrm{MCESP}(K_{n,k}, G)$ is solvable in polynomial time.*

PROOF. If we map one star of G in the graph $K_{n,k}$ we may remove that star from G and from $K_{n,k}$, yielding a graph $K_{n-i,k-j}$ where $i + j$ is the number of vertices in the star, after the removal we get a smaller instance of the same problem. We still need to find the value added to γ by that partial mapping, if the central vertex of the star was mapped to some vertex on the k side of $K_{n,k}$, then we added i edges to γ, if we mapped the central vertex to some vertex on the n side, we added j edges to γ. Since we are maximizing, that value will be $\max\{i, j\}$ if the star has at least one vertex on each side, and 0 otherwise, we will note this value m_{ij}. We are now ready to write the recursive expression that yields the algorithm.

$$\gamma(K_{n,k}, S_{n_1} \cup \cdots \cup S_{n_t}) = \max\{\gamma(K_{n-j,k-i}, S_{n_1} \cup \cdots \cup S_{n_{t-1}}) + m_{ij} : \\ i + j = n_t + 1 \text{ and } i, j \geq 0\}.$$

The fact that $\gamma = \Gamma$ follows directly by induction in t. Observe that there are $\mathcal{O}(nk)$ complete bipartite subgraphs of $K_{n,k}$, where the equality is taken as isomorphism. Also there are $t \leq n+k$ stars, and each star S_l admits at most $l+1 \leq n+k$ different configurations. Therefore it is enough to use a memory space of $\mathcal{O}\left(nk(n+k)^2\right)$ for memoization of partial results, and compute each entry at most one time, therefore this algorithm is polynomial. ■

CHAPTER 4

Complexity over additional graph classes

In this chapter we explore more restrictions to both graphs. In Section 4.1 we analyze grid-like graphs based on some existing ideas taken from [12]. We show MCESP is $\mathcal{NP}$-complete when one graph is a grid and the other is a union of grids, this holds for grids with 4, 6 and 8 neighbors. This also holds for honeycomb grids.

In the remaining part of the chapter we explore the cases when both graphs are restricted to split graphs, conntected proper interval graphs, trees and unions of paths. We also explore the case when the first graph is bipartite and the second a complete bipartite with isolated vertices. For all of these restrictions we prove that MCESP is $\mathcal{NP}$-complete.

4.1. Grids

As we stated in Section 1.1, the MCESP was first introduced in [4] as a formulation for a practical problem on array processors. Array processors are basically a set of partially interconnected processors, not every pair of processors are connected because of the high growth of links required, which is $\mathcal{O}(n^2)$ where n is the number of processors. When a pair of processors are connected they can share information faster than using the general bus. If we are given a set of programs to run in parallel and assuming we know how these programs share information, we are interested in using in the most convinient way the communication lines between processors. We may represent the array processor with a graph, having one vertex per processor and an edge between two vertices if and only if there is a communication line between the associated processors. In a similar way me can represent the communication between programs, one vertex per program and an edge between two vertices if and only if the associated programs share information. Clearly, if there are more programs than processors we may not map programs onto processors, so we assume that we have enough processors. Furthermore, if we have more processors than programs, we can add “dummy” programs that do nothing and communicate with no other program. Now it should be clear that solving MCESP for these two graphs is finding the best possible usage of communication lines.

In array processors a common configuration of communication lines has a grid-like structure. Some complexity results were suggested in [12], in the following we give the complete proofs based on their ideas. For this purpose we first introduce some definitions.

Definition 4.1.1. *A* 4-neighbor *grid graph* $G_{k,s}$ *of* k *rows and* s *columns is the graph given by* $P_k \times P_s$. *If* $V(P_k) = \{v_1, \dots, v_k\}$ *and* $V(P_s) = \{u_1, \dots, u_s\}$, *then the* i*th row of* $G_{k,s}$ *is the subgraph induced by* $\{(v_i, u_j) : 1 \leq$

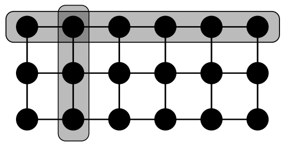

FIGURE 4.1.1. An example of a 4-neighbor grid graph of size 3×6, $G_{3,6}$.

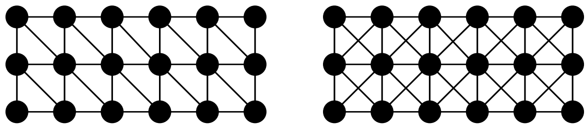

FIGURE 4.1.2. A 6-neighbor grid graph (a) and an 8-neighbor grid graph (b), both of size 3×6.

$j \leq s\}$, and the jth column is the subgraph induced by $\{(v_i, u_j) : 1 \leq i \leq k\}$. An example is given in Figure 4.1.1.

Definition 4.1.2. *A* 6-neighbor *grid graph $G_{k,s}$ is a 4-neighbor grid graph extended with the edges $((v_i, u_j), (v_{i+1}, u_{j+1}))$ for $1 \leq i \leq k-1$ and $1 \leq j \leq s-1$. An example is shown in Figure 4.1.2.*

Definition 4.1.3. *An* 8-neighbor *grid graph $G_{k,s}$ is a 6-neighbor grid graph extended with the edges $((v_i, u_j), (v_{i-1}, u_{j-1}))$ for $2 \leq i \leq k$ and $2 \leq j \leq s$. An example is shown in Figure 4.1.2.*

Observation 4.1.1. *A 4-neighbor grid graph $G_{k,s}$ has $2sk - s - k$ edges.*

Theorem 4.1.1. *The* MCESP *is $\mathcal{NP}$-complete when one graph is a 4-neighbor grid and the second a union of 4-neighbor grids.*

PROOF. The idea of the proof was given in [12], here we formalize the details using the proposed reduction. We reduce from 3-PARTITION [8]. The 3-PARTITION problem is, given an integer B and a multiset $A = \{a_1, \ldots, a_{3m}\}$ of integer numbers such that $B/4 < a_i < B/2$ and $\sum_{i=1}^{3m} ai = mB$, we are to decide if A be partitioned into m disjoint multisets $S_1, \ldots, S_m$ such that $\sum_{a \in S_i} a = B$. This problem is $\mathcal{NP}$-complete even if B is bounded by a polynomial on m [8]. Let $a^* := \min A$, and $k := \lceil (2m+1)/a^* \rceil$. Let $H := G_{2m,kB}$ and $G := \bigcup_{i=1}^{3m} G_{2,ka_i}$. Observe that $n_G = \sum_{i=1}^{3m} 2ka_i = 2kmB = n_H$, also note that k is bounded by a polynomial in m, hence the sizes of G and H are bounded by a polynomial in m. In the following we prove that $\Gamma(G, H) \geq m_G$ if and only if it is a YES 3-PARTITION instance.

Suppose there is a partition $S_1, \ldots, S_m$ of A such that S_i contains exactly 3 elements and each S_i sums B. For each $1 \leq i \leq m$ denote $S_i = \{a_{i1}, a_{i2}, a_{i3}\}$, map to the $(2i-1)$th and $2i$th row of H the graphs $G_{2,ka_{i1}}$, $G_{2,ka_{i2}}$ and $G_{2,ka_{i3}}$ secuentially, such that the mapping restricted to those vertices contributes all the edges of $G_{2,ka_{i1}}$, $G_{2,ka_{i2}}$ and $G_{2,ka_{i3}}$. Obviously this part of the mapping contributes $\sum_{j=1}^{3} 4ka_{ij} - 2 - ka_{ij}$. An

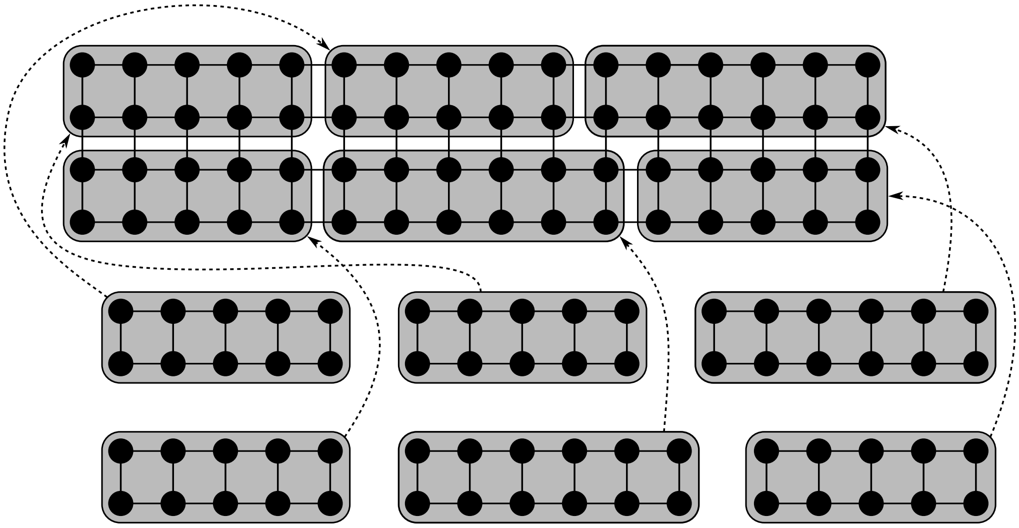

FIGURE 4.1.3. A mapping yielding m_G edges. The top graph is H and the bottom is G, the 3-PARTITION instance is given by $B = 16$, $A = \{5, 5, 5, 5, 6, 6\}$, one possible solution is $\{5, 5, 6\}, \{5, 6, 5\}$. Observe that $k = \lceil 4 + 1/5 \rceil = 1$.

example of this mapping is shown in Figure 4.1.3. This mapping yields $\gamma_f = \sum_{a \in A} 4ka - 2 - ka = m_G$.

To prove the converse we show that the connected components of G are arranged by the optimal mapping in double rows of H, afterwards we show why each such a row has exactly 3 subgraphs of G mapped onto it. We close the proof showing that by taking sets with the a_i's associated to those three subgraphs yields a YES certificate for 3-PARTITION.

Suppose $\Gamma(G, H) \geq m_G$, then G must be a subgraph of H. The width of each connected component of G is greater than the height of H, formally $ka^* \geq 2m + 1$, thus the component cannot be mapped vertically on H. Furthermore, the mapping must be "straight", otherwise we must have a vertex of degree at least 4 in G, which does not happen. Therefore, each connected component of G must be mapped horizontally. All the subgraphs are mapped in double rows, and each of these double rows is given by rows $2i - 1$ and $2i$ for $1 \leq i \leq m$, to see this observe that each connected component of G is a subgraph of H, and each vertex of H is part of some connected component of G via the optimal mapping. If we have a connected component mapped to the rows $2i$ and $2i + 1$, there must be another connected component of G that does not contribute all of its edges to Γ, this is a contradiction.

Finally we need to prove that for $1 \leq i \leq m$ exactly three connected components of G are assigned to vertices of rows $2i - 1$ and $2i$ in H. Suppose there are c connected components mapped to the subgraph induced by rows $2i - 1$ and $2i$, suppose these components are associated to the elements $\{a_{i_1}, \ldots, a_{i_c}\}$. If $c > 3$ then there are $\sum_{j=1}^{c} 4ka_{i_j} - 2 - ka_{i_j}$ edges contributing in the rows $2i-1$ and $2i$ of H. Observe that

$$\sum_{j=1}^{c} 3ka_{i_j} - 2 > \sum_{j=1}^{c} 3k\frac{B}{4} - 2 = 3kB\frac{c}{4} - 2c \geq 3kB - 2c.$$

Also note that at least $2(c-1)$ edges from the rows $2i-1$ and $2i$ are neccesarily lost due to the separation of the c connected components. Therefore the available edges on the $2i-1$ and $2i$th rows are $3kB-2-2(c-1) = 3kB-4-2c$, and we already saw that the contributing edges are more than $3kB-2c$, this is a contradiction, thus $c \leq 3$. Finally, since there are m double rows and $3m$ connected components in G and G is a subgraph of H, every double row of H must contain exactly 3 connected components of G via the optimal mapping.

For $1 \leq i \leq m$ construct the multisets $A_i := \{a_{i1}, a_{i2}, a_{i3}\}$, such that for $1 \leq j \leq 3$, $G_{2,ka_{ij}}$ is mapped to the double row given by rows $2i-1$ and $2i$ of H, this is well defined due to above observations of the optimal mapping. We have $\sum_{a\in A_i} ka = kB$, then $\sum_{a\in A} a = B$. Finally since the mapping is 1-1, the multisets $A_1, \ldots, A_m$ form a partition of A, thus, $A_1, \ldots, A_m$ is a positive certificate for 3-Partition. ■

Observation 4.1.2. *A similar result is shown in Theorem 4.6.1, but note that here one graph is neccesarily connected, thus we cannot use a trivial reduction to that $\mathcal{NP}$-complete problem.*

Corollary 4.1.1. *The* MCESP *is $\mathcal{NP}$-complete when one graph is a 6-neighbor grid and the second a union of 6-neighbor grids.*

Proof. The essence of the proof is the same as the proof of Theorem 4.1.1. We reduce again from 3-Partition and we encode the problem in graphs G and H in the same way as before, but using 6-neighbor graphs. Clearly this gives us a valid MCESP instance since the amount of nodes is the same as before in each graph. We again ask if G is a subgraph of H. We only need to observe that, when proving the second implication, i.e. if G is a subgraph of H then it is a Yes instance for 3-Partition, the connected components of G are mapped horizontally on H. We already saw that a connected component of G cannot be mapped vertically on H, because its longer than the height of H. Hence we need to analyze what happens when a connected component of G is mapped partially vertically and partially horizontally, and contributing all its edges. But if this is the case, then one connected component of G must have a vertex of degree at least 6, which is a contradiction. ■

Corollary 4.1.2. *The* MCESP *is $\mathcal{NP}$-complete when one graph is a 8-neighbor grid and the second a union of 8-neighbor grids.*

Proof. Same as in the above corollary. ■

The idea of above proofs may be used for many grid-like graphs, for example for honeycomb grids. A *honeycomb* grid is a set of "pasted" C_6 graphs as shows the Figure 4.1.4.

Corollary 4.1.3. *The* MCESP *is $\mathcal{NP}$-complete when one graph is a honeycomb grid and the second a union of honeycomb grids.*

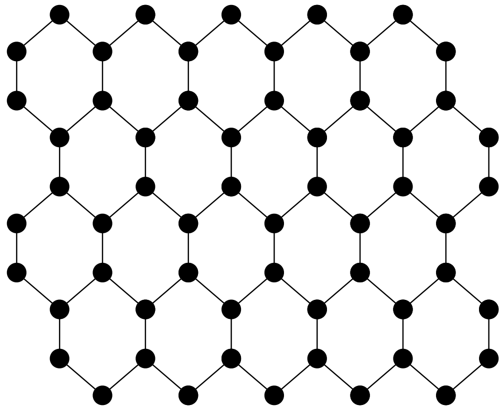

FIGURE 4.1.4. A honeycomb grid of 4 rows and 5 columns.

4.2. Complete Bipartite union isolated vertices vs. Bipartite graph

Definition 4.2.1. *The maximum edge biclique problem (* MPB *) asks for the maximum complete bipratite subgraph of G, the decision version is to decide if G contains a complete bipartite subgraph of k or more edges.*

Proposition 4.2.1. *The* MCESP *is $\mathcal{NP}$-complete when one graph is a complete bipartite union isolated vertices and the second a bipartite graph.*

PROOF. In [14] the authors proved that MBP is $\mathcal{NP}$-complete for bipartite graphs. We will prove MBP $\leq_P$ MCESP. Let $I := (G, k)$ be an instance of MBP with G a bipartite graph. For $0 \leq i \leq |V_G|$, $0 \leq j \leq i$ construct the MCESP instances

$$F = \{(K_{j,i-j} \cup \overline{K_{|V(G)|-i}}), G, j(i-j))\}.$$

Observe that MCESP$(K_{j,i-j} \cup \overline{K_{|V(G)|-i}}), G, j(i-j))$ is YES if and only if G contains a biclique of size $j(i-j)$. Therefore if we have MCESP$(f) =$ YES for some $f = (H, G, t) \in F$ such that $t \geq k$, then we have a biclique of edge size greater or equal to k in G. Reciprocally if we have a biclique of size $t \geq k$ in G, we can take all the instances of F with $K_{s,l} \cup \overline{K_{|V(G)|-s-l}}$ where $s + l = t$, and for some of them the MCESP answer must be YES. ∎

4.3. Split vs. Split graph

Proposition 4.3.1. *The* MCESP *is $\mathcal{NP}$-complete when we restrict both inputs to split graphs.*

PROOF. We reduce from MBP defined in Definition 4.2.1, this problem was proved to be $\mathcal{NP}$-complete for bipartite graphs in [14]. Let $G := (V_1 \dot\cup V_2, E)$ a bipartite graph with V_1 and V_2 independent sets, and $k \in \mathbb{N}$, we are to decide if there is a complete bipartite subgraph of edge size at least k in G.

Define a graph $G' := (V', E')$ with $2n$ vertices by extending G in the following way, $V' := V_1^+ \dot\cup V_2^+$ where $V_1 \subset V_1^+$, $V_2 \subset V_2^+$ and $|V_1^+| = |V_2^+| = n$, add all the possible edges between vertices of V_1^+ and all the edges between

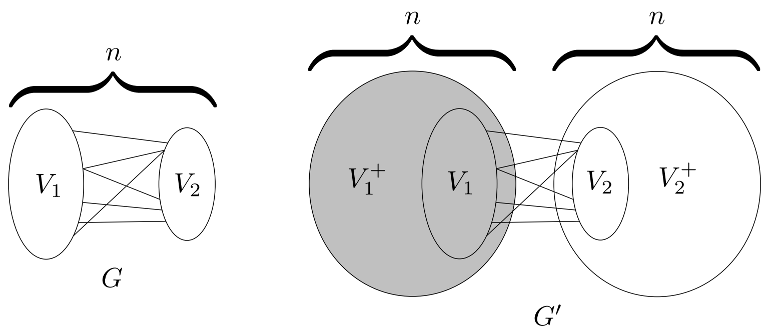

FIGURE 4.3.1. Construction of split graph G' from bipartite graph G. The shaded area represents the complete K_n subgraph.

V_1 and V_2 that appear in G. This construction is illustrated in Figure 4.3.1. It is clear that G' is a split graph, where V_1^+ induces a complete graph and V_2^+ is an independent set.

For each $k \leq l \leq n^2$ consider $H^l := (W_1 \dot\cup W_2, E_H)$ where W_1 induces a complete graph of size n, W_2 an independent set of size n, and there are l edges from W_1 to W_2 such that if we remove the edges from W_1 we get a complete bipartite graph of edge size l union isolated vertices. It must be noticed that we may obtain l edges with different partitions, and for a fixed l we have as many partitions as product decompositions of l in two factors, since there are at most $\mathcal{O}(l)$ different decompositions for l, there are at most $\mathcal{O}(l)$ different (non isomorphic) graphs. To distinguish these graphs we introduce the index j to H^l_j, for an example see Figure 4.3.2. This construction yields $\mathcal{O}(\sum_{l=k}^{n^2} l) = O(n^4)$ possible graphs H^l_j. Consider the family $\mathcal{I} := \{(G', H^l_j, n(n-1)/2 + l)\}$ of MCESP instances. In the following we prove that there is a YES instance $I \in \mathcal{I}$ for MCESP if and only if G contains a complete bipartite subgraph of edge size at least k.

Suppose we have a complete bipartite subgraph in G of edge size $l \geq k$ with l_1 vertices in V_1 and l_2 in V_2, observe that $l_1 l_2 = l$. Since $l \geq k$ there is a graph $H^l_j = (W_1 \dot\cup W_2, E_H)$ such that if we remove all the edges from W_1 we get a complete bipartite subgraph with l_1 vertices in W_1 and l_2 in W_2. Denote $V_1^+ = \{v_1, \dots, v_{l_1}, v_{l_1+1}, \dots, v_n\}$ where $v_1, \dots, v_{l_1+1}$ are the vertices of the complete bipartite subgraph of edge size l in V_1, and $V_2^+ = \{u_1, \dots, u_{l_2}, u_{l_2+1}, \dots, u_n\}$ with $u_1, \dots, u_{l_2}$ the vertices of the same complete bipartite subgraph. Also note $W_1 = \{a_1, \dots, a_{l_1}, a_{l_1+1}, \dots, a_n\}$ with $a_1, \dots, a_{l_1}$ the vertices of highest degree of W_1, and

$$W_2 = \{b_1, \dots, b_{l_2}, b_{l_2+1}, \dots, b_n\}$$

with $b_1, \dots, b_{l_2}$ the vertices with highest degree of W_2. Define $f : V_1^+ \dot\cup V_2^+ \longrightarrow W_1 \dot\cup W_2$ such that $f(v_i) = a_i$ for $1 \leq i \leq n$ and $f(u_r) = b_r$ for $1 \leq r \leq n$. Clearly $f(V_1^+) = W_1$, since V_1^+ and W_1 induce complete subgraphs and both have n vertices, these edges contribute $n(n-1)/2$ to γ_f. Furthermore all the edges with one endpoint in W_1 and the other in W_2 are covered by f by

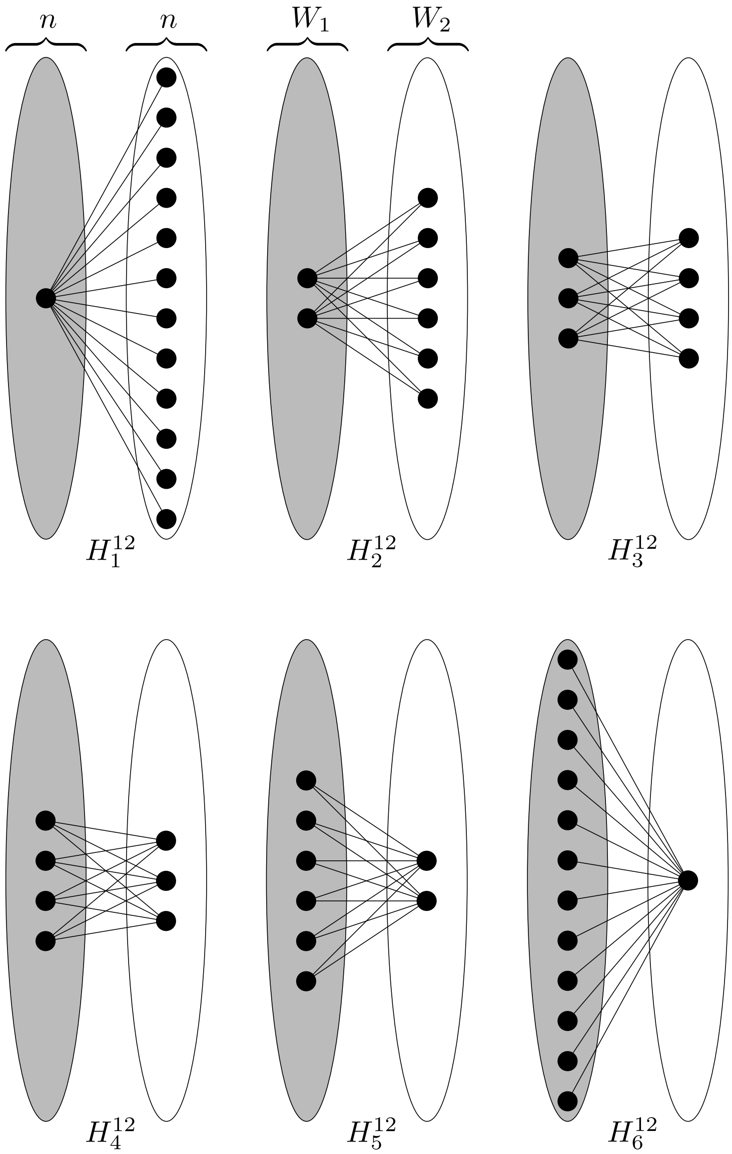

FIGURE 4.3.2. Example of H_j^l family for $l = 12$. The shaded area for each graph H_j^{12} is a K_n subgraph.

definition, these contribute l to γ_f, hence

$$\Gamma(G', H) \geq \gamma_f = n(n-1)/2 + l \geq n(n-1)/2 + k.$$

Suppose we have a YES instance $I \in \mathcal{I}$, then there is a graph H_l^j such that $\Gamma(G', H_j^l) \geq n(n-1)/2 + l$, denote $f : V(G') \longrightarrow V(H)$ the optimal 1-1 mapping. Since the graph H_l^j has $n(n-1)/2 + l$ edges then it must be a subgraph of G'. If we remove all the edges in V_1^+ from G', remove all the edges from W_1 in H and use the mapping f on these new graphs we obtain a complete bipartite subgraph in G' of size l. Since we did not add any edge between V_1^+ and V_2^+ except those in G, this complete bipartite subgraph of size $l \geq k$ is a subgraph of G. ∎

4.4. Proper Interval vs. Proper Interval Graphs

A *proper interval graph* G is an interval graph that admits a model without interval inclusions.

Proposition 4.4.1. *The* MCESP *is $\mathcal{NP}$-complete for connected proper interval graphs.*

Proof. The subgraph isomorphism problem is $\mathcal{NP}$-complete for connected proper interval graphs when both graphs have the same number of vertices, this result is shown in [9]. Given G and H connected proper interval graphs with $n_G = n_H$, deciding if H is a subgraph of G is the same as deciding if $\Gamma(G, H) \geq m_H$. ∎

4.5. Tree vs. Tree

In [1] the authors proved that MCESP is $\mathcal{NP}$-complete when both graphs are trees. For completion we reproduce their proof here.

Theorem 4.5.1. MCESP *is $\mathcal{NP}$-complete when both input graphs are trees.*

Proof. We reduce from 3-Partition [8]. Let $B \in \mathbb{N}$, and a multiset $A = \{a_1, \ldots, a_{3m}\}$ of integers such that $B/4 < a_i < B/2$ for $1 \leq i \leq 3m$ and $\sum_{i=1}^{3m} a_i = mB$, an instance of 3-Partition.

A *spider* is a tree with exactly one node of degree greater than two. The paths extending from the high-degree center are called hairs. Let T_1 be a spider with m hairs, each with $B + 3$ edges. Let T_2 be an extended star with $3m$ hairs, where the ith hair has $a_i + 1$ edges. Note that both T_1 and T_2 have $m(B + 3)$ edges, T_1 has $m(3 + B) + 1$ vertices, and T_2 has $1 + \sum_{a_i \in A} 1 + a_i = 1 + m(3 + B)$ vertices.

A 3-partition of A yields a common subtree on all the vertices of T_1 with only $2m$ edges missing, or 2 per hair. We claim that any common subtree must miss at least $2m$ edges. Suppose that only one edge was missing on a given hair in T_1. The larger remainder of the hair must contain at least $B/2$ edges, which can then only be matched to two hairs of T_2 including the root. Hence, $3m - 2$ of the edges incident to the root of T_2 must be eliminated. Thus, there is a common subtree missing only two edges per hair in T_1 if and only if there is a 3-partition of A. ∎

4.6. Union of Paths vs. union of Paths

Theorem 4.6.1. *The* MCESP *is $\mathcal{NP}$-complete when both graphs are restricted to union of paths.*

Proof. We reduce from 3-Partition. Let $B \in \mathbb{N}$, and a multiset $A = \{a_1, \ldots, a_{3m}\}$ of integers such that $B/4 < a_i < B/2$ for $1 \leq i \leq 3m$ and $\sum_{i=1}^{3m} a_i = mB$, an instance of 3-Partition. Construct the graphs $G := \bigcup_{i=1}^{3m} P_{a_i+1}$ and $H := \bigcup_{i=1}^{m} P_{B+3}$, observe that $n_G = \sum_{i=1}^{3m} a_i + 1 = mB + 3m = \sum_{i=1}^{m} B + 3 = n_H$. In the following we prove that $\Gamma(G, H) \geq mB$ if and only if it is a Yes 3-Partition instance.

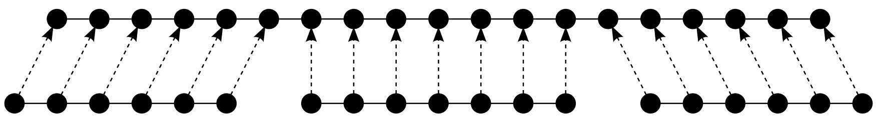

FIGURE 4.6.1. Part of the mapping f, the dotted arrows shows where a vertex is mapped. The P_{19} belongs to H and the rest of the paths to G.

Suppose there is a partition $S_1, \ldots, S_m$ of A such that S_i contains exactly 3 elements and each S_i sums B. For each $1 \leq i \leq m$ denote $S_i = \{a_{i1}, a_{i2}, a_{i3}\}$, map $P_{a_{i1}+1}$ on the first $a_{i1} + 1$ vertices of the ith P_{B+3} of H such that the mapping contributes a_{i1} edges to γ_f. Then map $P_{a_{i2}+1}$ to the ith P_{B+3}, beggining at the vertex $a_{i1} + 2$, and ending at the vertex $a_{i1} + a_{i2} + 2$, such that this contributes a_{i2} edges to γ_f. Finally map $P_{a_{i3}+1}$ to the remaining vertices of the ith P_{B+3} contributing a_{i3} edges to γ_f. Then $\gamma_f = \sum_{i=1}^{3m} a_i = mB$, because all the edges from G are contributing to γ_f. By definition $\Gamma(G, H) \geq \gamma_f$. An example is illustrated in Figure 4.6.1.

Suppose there is a 1-1 mapping $f : V_G \longrightarrow V_H$ such that $\gamma_f \geq mB$. Since $m_G = mB$, G must be a subgraph of H, then each P_{a_i+1} is mapped via f in such way that all the vertices are contiguous. Take any P_{B+3} subgraph of H, we have k different P_{a_i+1} subgraphs of G mapped onto it, label them $P_{a_{i_1}+1}, \ldots, P_{a_{i_k}+1}$. If $k > 3$ then there are $\sum_{j=1}^{k} a_{i_j}$ edges contributing from P_{B+3}. Observe that

$$\sum_{j=1}^{k} a_{i_j} > \sum_{j=1}^{k} \frac{B}{4} = \frac{kB}{4} \geq B.$$

Also note that $k - 1 \geq 3$ edges from P_{B+3} are neccesarily lost due to the separation of different $P_{a_{i_j}+1}$. Since P_{B+3} has exactly $B + 2$ edges, and at least 3 are lost, we cannot contribute more than $B - 1$ edges with P_{B+3} when $k > 3$, hence $k \leq 3$. On the other hand, if $k < 3$ we have that

$$B + 2 - \sum_{j=1}^{k} a_{i_j} > B + 2 - k\frac{B}{2} \geq 2$$

edges are not contributing from P_{B+3}. Stated in other words, we lose at least 3 edges in that P_{B+3}. Observe that if $k = 3$, then we lose at least 2 edges from P_{B+3} due to separation, and we already saw that no P_{B+3} may have $k > 3$, therefore, if we lose 3 edges in one P_{B+3} we cannot "balance" the contributions over the rest of the paths in H, yielding $\gamma_f < mB$, which is a contradiction. Therefore k is always 3 if $\gamma_f \geq mB$, furthermore, each of the paths in H loses exactly 2 edges. Define $S_1, \ldots, S_m$ as $S_i := \{a_j : P_{a_j+1}$ is mapped to the ith $P_{B+3}\}$, since $k = 3$ for each P_{B+3} we get that each S_i has exactly 3 elements, since exactly 2 edges are lost in each P_{B+3}, each S_i sums B, finally, since f is a 1-1 mapping, $S_1, \ldots, S_m$ is a disjoint partition of A, thus it is a positive certificate for 3-Partition.

■

CHAPTER 5

The End

5.1. Conclusions

Our main goal was to study the time complexity behavior of MCESP under different restrictions, aiming to obtain a classification into $\mathcal{NP}$-complete and polynomial cases. From the the beggining we expected to verify that MCESP remains $\mathcal{NP}$-hard in many strong restrictions. In Chapter 3 and Chapter 4 we explored different restrictions and some of them were strong enough to verify the initial thoughts. Nevertheless we found some polynomial cases where we did not expect them, for example when one graph is complete bipartite and the second a cograph, this result is detailed in Section 3.3. In the following two paragraphs we give a summary of analyzed restrictions.

In Chapter 3 we worked with one graph restricted to complete bipartite class. When the second graph belongs to one of the following families

- arbitrary graphs
- split graphs
- chordal graphs (follows directly since split graphs are chordal graphs)
- tripartite graphs
- co-bipartite graphs

then the MCESP is $\mathcal{NP}$-complete. On the other hand, if the second graph belongs to

- complete bipartite graphs
- cographs
- union of stars

then the problem is polynomial. We were not able to classify the case when the second graph is a bipartite graph, although some observations of this restriction were made in Section 3.4, based on those observations we think this case is $\mathcal{NP}$-hard.

In Chapter 4 we explored more restrictions to both graphs. In Section 4.1 we analyzed grid-like graphs based on some existing ideas, proving that MCESP is $\mathcal{NP}$-complete when one graph is a grid and the other is a union of grids, this holds for grids with 4, 6 and 8 neighbors. We also noticed that those ideas may be adapted for the analysis of other grid-like graphs, like honeycomb grids. In the remaining part of Chapter 4 we showed that MCESP is $\mathcal{NP}$-complete when the restrictions are

- both graphs are split graphs
- both graphs are connected proper interval graphs
- both graphs are trees
- both graphs are unions of paths

- one is complete bipartite graph union isolated vertices and the other a bipartite graph.

While analyzing different aspects of the problem we found some general results related to MCESP, which were not intensively used to contribute to the classification of restrictions, therefore we arranged those results in Chapter 2. Here we observed how the MCESP behaves when one graph is complemented and related it with the MinCESP, thus relating the complexity of both problems. Using this relation of problems, in Observation 2.1.1 we mentioned that MCESP is $\mathcal{NP}$-complete for two graph classes if and only if it is $\mathcal{NP}$-complete when restricted to the complement of those classes. We gave a possible graph distance definition using the MCESP in Section 2.2. Finally in Section 2.3 we related the MCESP with Graph-Isomorphism and formalized why the MCESP is in fact a generalization of Graph-Isomorphism problem, and not only of the Subgraph-Isomorphism problem.

5.2. Further Work

The results shown in Chapter 2 are general enough to become a first step for researching how the structure of the mapping behaves under different restrictions. For example, in Section 2.2 we saw that restricting graphs to have the same number of edges gives a notion of a distance over such graph class, we think that restricting to other classes may lead to interesting and unexpected results, relating the MCESP with other problems. Finally, we do not expect interesting results from relating the Graph-Isomorphism problem and MCESP, intuitively one may think that the relations are somehow generalizations of the relations between Graph-Isomorphism and Subgraph-Isomorphism. Although a more theoretical approach may be chosen, and explore what kind of relations may be found with the polynomial hierarchy, we did not studied this field in this work, but there might be some relations.

In Chapter 3 some observations were made when one graph is restricted to complete bipartite and the other to an arbitrary bipartite, we think this is an $\mathcal{NP}$-hard case, the reason to believe this is the reformulation shown in Lemma 3.4.1. Despite our beliefs we were not able to prove this, an interesting work is to search for such a proof.

In Chapter 3 and Chapter 4 we classified some restrictions of MCESP as polynomial and others as $\mathcal{NP}$-hard. For the $\mathcal{NP}$-hard cases a more refined analyisis would be a classification in approximable or non approximable assuming $\mathcal{P} \neq \mathcal{NP}$. If a restriction turns out to be approximable, searching the best possible approximation factor is an interesting work. This is, probably, the most practical classifcation in the $\mathcal{NP}$-hard class. Another interesting analysis is to classify whether an $\mathcal{NP}$-hard restriction is Fixed Parameter Tractable or not.

Bibliography

[1] T. Akutsu and M. M. Halldórsson. On the approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233(1–2):33 – 50, 2000.

[2] V. Arvind and P. P. Kurur. Graph isomorphism is in spp. *Information and Computation*, 204(5):835 – 852, 2006.

[3] H. L. Bodlaender and K. Jansen. On the complexity of the maximum cut problem. In P. Enjalbert, E. W. Mayr, and K. W. Wagner, editors, *STACS*, volume 775 of *Lecture Notes in Computer Science*, pages 769–780. Springer, 1994.

[4] S.H. Bokhari. On the mapping problem. *IEEE Transactions on Computers*, 30(3):207–214, 1981.

[5] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[6] D.G. Corneil, H. Lerchs, and L. S. Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163 – 174, 1981.

[7] S. Földes and P.L. Hammer. Split graphs. In *8th South-Eastern Conf. on Combinatorics, Graph Theory and Computing*, Congressus Numerantium 19, pages 311–315. (F. Hoffman et al. eds.) Louisiana State Univ., Baton Rouge, Louisiana, 1977.

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

[9] S. Kijima, Y. Otachi, T. Saitoh, and T. Uno. Subgraph isomorphism in graph classes. *Discrete Mathematics*, 312(21):3164 – 3173, 2012.

[10] J. Köbler, U. Schöning, and J. Torán. Graph isomorphism is low for pp. *COMPUT. COMPLEXITY*, 2:301–330, 1992.

[11] L. A. Levin. Universal search problems (Универсальные задачи перебора). *Problems of Information Transmission (Проблемы передачи информации)*, 9(3), 1973.

[12] J. Marenco. Un algoritmo branch and cut para el problema de mapping. *Departamento de Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires*, Tesis de licenciatura, 1999.

[13] B. D. McKay. Practical Graph Isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[14] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[15] D. C. Schmidt and L. E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM*, 23(3):433–445, July 1976.

[16] U. Schöning. Graph isomorphism is in the low hierarchy. In F. J. Brandenburg, G. Vidal-Naquet, and M. Wirsing, editors, *STACS*, volume 247 of *Lecture Notes in Computer Science*, pages 114–124. Springer, 1987.