University of Ljubljana
Faculty of computer and Information Science

Aleksander Fabijan

Online Algorithms for Graph Partitioning into Cliques

DIPLOMA THESIS

UNIVERSITY STUDY PROGRAMME COMPUTER AND INFORMATION SCIENCE

Mentor: dr. Andrej Brodnik

Co-mentor: prof. dr. Bengt J Nilsson

Ljubljana 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksander Fabijan

Sprotni algoritmi za računanje razdelitve grafa na klike

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM RAČUNALNIŠTVO IN INFORMATIKA

Mentor: dr. Andrej Brodnik

Somentor: prof. dr. Bengt J Nilsson

Ljubljana 2014

Faculty of Computer and Information Science issues the following thesis:

Thesis topic:

Search for the largest clique in a graph is a well known NP-complete problem. We generalise it not to search for the largest clique, but to partition graph in subgraphs, where each of them is a clique. The described problem is met in bioinformatics, when we calculate gene expression.

The described problem has a trivial solution where we partition graph into pairs of connected vertices each of the pair is obviously a clique. Define a more reasonable profit function that will guide a partitioning of the graph into a cliques and design an algorithm to perform such a partition. Algorithm shall work as an on-line algorithm, in which case the vertices with their adjacent edges are coming one at a time, while the algorithm has to compute an appropriate partition in each step. Estimate the time complexity of your algorithm, its quality and the complexity of the problem itself.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Problem iskanja največje klike v grafu je dobro poznani NP-polni problem. Problem posplošimo tako, da ne iščemo največje klike, ampak želimo razdeliti graf na podgrafe, ki so vsak zase klika. Opisani problem je primer problema, ki ga srečamo v bioinformatiki in rešuje problem iskanja izražanja genov.

Ta problem ima seveda trivialno rešitev, kjer povežemo po dve vozlišči grafa v kliko. Definirajte smiselnejšo cenovno funkcijo razdelitve grafa v klike in začrtajte algoritem, ki zagotavlja v čim večji meri takšno razdelitev. Algoritem naj deluje v sprotnem načinu dela, kar pomeni, da prihajajo vozlišča s svojimi povezavami eno za drugim in mora algoritem sproti naračunavati novo razdelitev. Ocenite časovno zahtevnost algoritma, kakovost njegove rešitve in časovno zahtevnost problema.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Aleksander Fabijan z vpisno številko **63080086**, sem avtor diplomskega dela z naslovom:

Sprotni algoritmi za računanje razdelitve grafa na klike

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom Dr. Andreja Brodnika in somentorstvom prof. Dr. Bengt J Nilsson,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki ”Dela FRI”.

V Ljubljani, dne 15. avgusta 2014 Podpis avtorja:

I would like to say thank you to Dr. Andrej Brodnik who offered me all the support I needed and the opportunity to cooperate with Dr. Bengt J. Nilsson and Dr. Mia Persson from Malmö University, Sweden. Thanks to my brother, who encouraged me to accept this challenge and, last but not least, to my parents who supported me during my studies.

To my brother, Silvester.

Contents

List of abreviations

Abreviations	**Full Form**
MAX-ECP	Maximum Edge Clique Partition Problem
MIN-ECP	Minimum Edge Clique Partition Problem
PTAS	Polynomial-Time Approximation Scheme
APX-hard	Approximable-hard
NP-hard	Non-deterministic Polynomial-time hard

Abstract

Clique clustering is the problem of partitioning a graph into cliques so that some objective function is optimised. In online clustering the input graph is given one vertex at a time, and vertices that have been previously clustered are not allowed to be separated. The objective is to maintain a clustering that never deviates too far from the optimal offline solution.

We give a constant competitive upper bound and a strategy (Lazy) for online clique clustering, where the objective function is to maximise the number of edges inside the clusters (Max-ECP). We also give almost matching upper and lower bounds on the competitive ratio for online clique clustering, where we want to minimise the number of edges between clusters (Min-ECP). In addition, we prove that the *greedy* method only gives linear competitive ratio for these problems.

The research result shows that the proposed constant competitive strategy performs significantly better on bigger graphs than the *greedy* method.

Results in this thesis have been published on 8th International Conference, CIAC 2013 in Barcelona, Spain. Scientific article Competitive Online Clique Clustering is available in the Proceedings on pages 221 until 233.

Keywords: competitive analysis, clustering, online algorithm, approximation algorithm

Povzetek

Grupiranje v klike je proces združevanja vozlišč v gruče, za katere velja, da so vsa vozlišča med seboj povezana. V sprotnem (*on-line*) združevanju celoten graf ni znan vnaprej, ampak je na voljo po eno vozlišče naenkrat. Tista vozlišča, ki so že pridružena gruči, ne morejo biti prestavljena v drugo gručo. Naloga je poiskati takšno razvrstitev vozlišč, ki se od optimalne razvrstitve razlikuje čim manj.

V tej diplomski nalogi podamo konstantno zgornjo mejo in algoritem (*Lazy*) za problem sprotnega združevanja v klike, kjer je cilj poiskati razvrstitev vozlišč s čim več povezavami znotraj gruč (problem Max-ECP). Poleg tega podamo ujemajoči zgornji in spodnji meji za problem sprotnega združevanja v klike, kjer je cilj poiskati razvrstitev s čim manj povezavami med gručami (problem Min-ECP). Za oba problema pokažemo, da naraven (*Greedy*) pristop vodi k linearni rešitvi.

Naša metoda *Lazy* nudi konstantno tekmovalno razmerje, kar se znatno odraža na grafih z veliko vozlišči.

Rezultati te diplomske naloge so bili objavljeni na osmi mednarodni konferenci, CIAC 2013 v Barceloni v Španiji. Članek Competitive Online Clique Clustering je na voljo od strani 221 do strani 233.

Ključne besede: analiza konkurenčnosti, grupiranje, sprotni algoritem, aproksimacijski algoritem

Daljši povzetek

V tej diplomski nalogi se bralec seznani s sprotnimi algoritmi. Ti predstavljajo ogrodje za raziskovanje problemov v računalništvu, kjer vsi podatki niso znani vnaprej, ampak so algoritmu, ki rešuje problem, na voljo postopoma. Pred kratkim so vzbudili veliko zanimanja zaradi številnih problemov, ki jih ni mogoče rešiti na drugačen način. Hkrati rešujejo probleme, za katere želimo poiskati približno rešitev, ker bi iskanje natančne potekalo predolgo. Mnogokrat se v praksi izkaže, da se natančne rešitve nekega problema ne da poiskati in da se moramo pogosto zadovoljiti s približno rešitvijo, če bistveno ne odstopa od natančne (optimalne) rešitve.

Teoretično računalništvo se velikokrat ukvarja z algoritmi in ocenjevanjem učinkovitosti nekega algoritma, v primerjavi z drugim. Strokovno rečemo takšnemu primerjanju uspešnosti različnih algoritmov analiza konkurenčnosti (*competitive analysis*). Ta ima pomembno vlogo pri tako imenovanih sprotnih algoritmih. Pri njih so vhodni podatki programu na voljo na tako imenovani zaporedni način. Analogija prihaja iz sredine šestdesetih let, ko so tako prvič poimenovali prenos podatkov preko serijskega načina - poslušalec je pasiven in začne izvajati program šele po prejemu prvega podatka. Na podlagi zadostne količine podatkov je takšen program nekoč posvetil z žarnico in poslušal dalje.

Pri sprotnih (*on-line*) algoritmih je izvajanje programa pogojeno s tem, koliko podatkov je znanih. Koraki, ki jih je program izbral na poti do trenutnega stanja močno vplivajo na pot, ki bo izbrana v prihodnje. Primer problema, ki nima znanega celotnega vhoda je sprotno prilagajanje poti v službo, kjer navigacijski sistem odloča, po kateri poti nas bo pripeljal do delovnega mesta. Algoritem v navigacijski napravi na podlagi podatkov o prometu, zastojih, delu na cesti in trenutnem položaju izračuna predlagano pot, ki pa se bo v času potovanja prilagodila ob morebitni spremembi razmer na cesti. Podobno, računalnik na delovnem mestu vedno skrbi, da v hitrem pomnilniku ostanejo tisti podatki, za katere obstaja velika verjetnost, da bodo kmalu zopet potrebni.

Omenjena primera pa sta le dva izmed mnogih, kjer se sprotni algoritmi uporabljajo za modeliranje procesov. Najdemo jih tudi pri časovnem načrtovanju opravil, kjer so naloge podane v zaporedju in morajo biti končane pred prihodom novih zahtev ter v omrežjih pri ohranjanju odprtih povezav kjer ni znano katere bodo uporabljene v prihodnje.

Posebno in pomembno vlogo pa imajo sprotni algoritmi v biotehniki in genetiki, kjer znanstveniki raziskujejo in modelirajo človeški genom. Genetiki pogosto merijo nivoje pri katerih se geni izločajo. Farmacevti npr. ugotavljajo, kako se stanje celice spreminja glede na to, koliko zdravila je bilo dodanega v biološki sistem [1]. Meritve se izvajajo v različnih pogojih in različnih tkivih. Vrednosti, pri katerih so doseženi pragovi so ključnega pomena pri razumevanju delovanja posameznih genov. Rezultati teh meritev pa predstavljajo vhodne podatke za sprotni algoritem, ki sestavlja model preučevanega gena ali celice.

V procesu preučevanja DNA so pod drobnogledom posamezni geni, ki z medsebojno interakcijo vplivajo na telesni razvoj in vedenje. Sestavljeni so iz verige DNK (pri nekaterih virusih RNK), ki vsebuje promotor za nadzor aktivnosti gena in kodirajoče zaporedje, ki določa rezultat gena. Kadar je gen aktiven, se kodirajoče zaporedje v procesu prevajanja (transkripcija) prepiše, pri čemer nastane kopija RNK v genu vsebovanih podatkov. Ta RNK lahko nato na podlagi genskega koda usmerja sintezo proteinov. Nekatere RNK pa se uporabijo neposredno, npr. kot deli ribosomov. Molekule, ki nastanejo z izražanjem genov, bodisi RNK ali proteini, so genski produkti.

Pomemben korak v analizi podatkov pridobljenih iz procesa izločanja genov je iskanje skupin, ki so sestavljene iz genov, ki so izločili podoben vzorec. Strokovni izraz za takšno skupino elementov je gruča in lastnost te je, da vsebuje elemente, ki so si med seboj podobni (homogenost). V primeru genetike so to skupine genov, za katere velja, da se enako obnašajo (imajo izmerjeno enako vrednost izločanja določenega proteina) ob enakih pogojih, npr. isti količini dodanega zdravila. Za proces grupiranja, gene predstavimo kot vozlišča v grafu G skupaj s povezavami, ki so prisotne med tistimi geni, ki imajo podoben vzorec izločanja proteinov. Združevanje v gruče je sestavljeno iz razdelitve grafa v polno povezane podgrafe tako, da zadovoljimo kriterijsko funkcijo. V tako imenovanem sprotnem načinu združevanja v gruče je vhodni graf razkrit po eno vozlišče naenkrat in vozlišča, ki so že bila združena v gručo ni dovoljeno razčleniti. Naloga je poiskati gruče, ki so čim bližje optimalni rešitvi.

Obstaja več različnih kriterijskih funkcij, ki so v uporabi; maksimiziranje števila povezav znotraj gruč + število nepovezav med gručami (MaxAgree), ali npr. minimiziranje števila nepovezav v gručah + število povezav med gručami (MinDisagree). Nekatere druge mere zahtevajo, da so vozlišča v gručah povezana med seboj (clique), kjer je naloga maksimizirati število povezav znotraj gruč ali minimizirati število povezav med gručami. S tujko rečemo primeru takšne kriterijske funkcije MAX-ECP (Maximal-Edge Clique Partition) in Min-ECP (Minimum-Edge Clique Partition).

Prva, Max-ECP je podrobno opisana v poglavju 2.1.1. Ta kriterijska funkcija je uporabna v področju izločanja genov in klasificiranju DNA-ja [2]. Bralec lahko najde v poglavju 4.2 požrešen pristop implementacije algoritma Max-ECP. Tak algoritem zadovolji potrebe kriterijske funkcije, vendar ni prav zelo učinkovit v primerjavi s strategijo, ki pozna vse podatke vnaprej (OPT). Izkaže se, da je $(\frac{2}{|V|-2})$-tekmovalen. Njegova slaba lastnost je, da je tekmovalno odvisen od števila vozlišč v grafu - $|V|$. Iz tega razloga najdemo v poglavju 4.3 optimizirano strategijo Lazy, ki je v primerjavi s tako imenovano strategijo off-line učinkovitejša. V slovenščini je algoritem Lazy podan na Sliki 1 in v grobem ločimo njegovo izvajanje med dvema vejama:

- Če optimalna rešitev OPT_n (optimalno gručenje nad n vozlišči) ne vsebuje gruč večjih kot 3 vozlišča, se strategija Lazy izvaja enako kot *greedy*.
- V nasprotnem primeru grupira Lazy vozlišča v samostojne gruče vse dokler razmerje profitov med trenutnim gručenjem S'_n in optimalnim (off-line) gručenjem OPT_n (nad n znanimi vozlišči) ni nad določenim pragom. Ko je prag dosežen, Lazy poračuna *relativen optimum* (optimalno gručenje nad razkritimi vozlišči in pripadajočimi povezavami).

Strategija Lazy ni neposredno odvisna od števila vozlišč v grafu in je $9/(154+16\sqrt{61}) \approx 0.032262$-tekmovalna, kar smo z uporabo matematike dokazali v poglavju 4.3.2. Dodatno je vredno omeniti, da število gruč ni znano vnaprej in je

```
Strategy  Lazy
   /* Drži $S_n$ z profitom profit($\mathrm{S_n}$) in naj bo $c$ konstanta */
1        $n = 1$
2        while novo vozlišče $v_n$ na vhodu do
2.1          $S'_n := S_{n-1} + singleton(v_n)$
2.2          Poračunaj $OPT_n$
2.3          if največja gruča v $OPT_n \geq 4$ then
2.3.1            if profit($\mathrm{OPT_n}$) $> c \cdot$ profit($\mathrm{S'_n}$) then
2.3.1.1              Izračunaj relativen optimum $S'_n$, $\widehat{OPT}(S'_n)$
2.3.1.2              Sestavi $S_n$ z uporabo $\widehat{OPT}(S'_n)$
                 else
2.3.1.3              $S_n := S'_n$
                 endif
             else
2.3.2            Sestavi $S_n$ z uporabo greedy strategije
             endif
2.4          $n := n + 1$
         endwhile
End  Lazy
```

Slika 1: Strategija Lazy

dinamično določeno med izvajanje strategije. Neposredna primerjava požrešne strategije in na drugi strani tako imenovane strategije Lazy pokaže, da je rezultat slednje precej bližje pravilnemu.

V poglavju 5 predstavimo spodnjo mejo za on-line Min-ECP združevanja v gruče in idejo požrešne strategije (5.2). Najprej predstavimo dokaz, da ne more obstajati deterministična strategije, ki bi rešila problem MIN-ECP in bila več kot $\frac{n^{1-\epsilon}}{2}$-tekmovalna. V nadaljevanju pokažemo, da je požrešna strategija $\frac{(n-2)}{2}$-tekmovalna in da pripada skupini algoritmov, ki smo jo poimenovali *Maksimalni.*

Chapter 1

Introduction

This thesis provides an insight into Online Algorithms. They represent a theoretical framework for studying problems in interactive computing and they gained a lot of attention recently because of the number of problems that cannot be solved offline or the desire to find approximate solutions to problems that seem to be too complex, perhaps even too difficult, to be solved otherwise.

We focus on online clique clustering, an online problem that belongs to a wider group of problems called correlation clustering and it has applications in Genetics and Biotechnology. A common activity there is to analyse the human genome. This is done by researchers who identify the levels at which a particular gene is expressed within a cell (gene expression for short), tissue or organism, and it can be of significant importance when finding viral infection of a cell, detecting vulnerability to cancer or finding out if a bacteria is resistant to penicillin or any other drug.

Recent advances in computer science and biotechnology allow scientists to measure the expression levels for multiple genes at the same time, producing a significant amount of data. Valuable information is extracted and analysed. Here an important step is to identify groups of genes that produce identical or similar levels of expression. This translates to the algorithmic problem of clustering. The detection of groups of genes that exhibit similar expression patterns is a key initial step in the analysis of gene expression data.

We provide more detailed insight on the problem in the next chapters.

1.1 Intuitive Definition of the Problem

Consider the following problem: Given an input sequence of vertices $A, B, C, D, ...$ that belong to a graph G, determine a clustering that maximises a given objective function. Moreover, the vertices and its associated edges to previous vertices are revealed in a serial fashion (one at a time) and the clustering algorithm has to be executed in every step. Upon an arrival of a new vertex, the later is either clustered into an already existing cluster using merge operation, or it is put into a singleton cluster. Existing clusters can be merged together, however, this operation is irreversible, hence a bad decision will have significant impact on the final solution.

The problem mentioned above belongs to a group of correlation clustering problems. Its different variants have been extensively studied over the past decades; see e.g. [5]. Several objective functions are used in the literature, e.g., maximise the number of edges within the clusters plus the number of non edges between the clusters (MaxAgree), or minimise the number of non-edges within the clusters plus number of edges between the clusters (MinDisagree).

Other measures require that the clusters are cliques, complete subgraphs of the original graph, in which case we maximise the number of edges inside the cluster or minimise the number of edges outside the clusters (not considering the edges between clusters). These measures gave rise to the maximum and minimum edge clique partition problems (Max-ECP and Min-ECP) respectively.

In this thesis we present Max-ECP and Min-ECP problems in their online version, simulate two algorithms to solve them (Greedy and Lazy) and prove what results the algorithms achieve.

1.2 Structure of the Work

This thesis is structured as following: First an overview of related and prior work is given in section 2, together with an introduction to clustering and different types of clustering (section 2.1) and definitions of criteria functions Maximum Edge Clique Partition and Minimum Edge Clique Partition (Max-ECP and Min-ECP).

First, an introduction to online algorithms is given in Section 2.2. Next, introduction to Graph Theory is given in Chapter 3. In Chapter 4 Online Max-ECP clustering is described in details. Two different approaches are described for the Max-ECP objective function. The first one shows that the *greedy* strategy is far from optimal and is no better than a $\frac{2}{|V|-2}$-competitive (Section 4.2). We give another strategy in Section 4.3. We named it Lazy and it is a Constant Competitive Strategy due to its property of being competitive to an offline version with a difference of a constant which we calculated to be ≈ 0.032262.

We discuss Online Min-ECP Clustering in chapter 5, presenting a lower bound in section 5.1 and a *greedy* strategy for Min-ECP problem in section 5.2.

Investigating a possibility of improving the competitiveness of both Min-ECP and Max-ECP would ensure better results and is therefore one of the tasks to be researched in the future (Chapter 6).

Chapter 2

Related and prior work

Analysing algorithms where the performance of an online algorithm is compared to the performance of an optimal offline algorithm (the one that has the complete knowledge of the input) was introduced by Sleator and Tarjan [12]. This type of analysis is called competitive analysis and they studied the amortised efficiency of the move-to-front algorithm (where after accessing or inserting an item in a list, the item is moved to the front of the list, without changing the relative order of the other items) for dynamically maintaining a linear list. They also analysed the amortised complexity of LRU (last recently used) algorithm, showing that its efficiency differs from the offline version of the paging rule by a factor that depends on the size of the fast memory. Competitive analysis requires that any algorithm performs well both on hard and easy inputs, where both are defined by the performance of the optimal offline algorithm. Competitive algorithms were also developed for distributed systems, where the algorithm has to react to a request arriving at a location on site B, without the knowledge of what the state of requests on site A is [13].

In this thesis we will focus on competitive analysis of algorithms that work with clustering. This is an important issue in the analysis and exploration of data and is considered as the most important unsupervised learning problem. It consists of discovering natural groups of similar elements (in our case these are verticies) in data sets. After the process is complete these data sets are called clusters.

2.1 Different types of Clustering

Clustering is a machine learning technique for analysing data and dividing it into groups of similar data. In our case, the data are the vertices of a graph. The groups or sets of similar data are known as clusters.

There are in general four types of division that each clustering algorithm can be categorised into [11]:

- Hierarchical or partitional: A hierarchical clustering algorithm is based on the union between the two nearest clusters. At the beginning, every example (in our case vertex) is put into a singleton cluster and using merge operations in the next iterations we achieve the final clustering. A hierarchical structure that reflects the order in which groups are merged or divided is created.
  In partitional clustering examples are divided into clusters and then evaluated by some criterion (e.g. Sum of Squared Error). Additionally, the number of clusters has to be given before the process of clustering can begin. Partitioning algorithms iteratively relocate objects (in our case vertices) among groups until criterium has convergent. The hierarchical structure does not reflect the order in which groups are merged. The most known partitional clustering algorithm is k-means, described in [11].
- Exclusive or overlapping: In exclusive clustering an example (in our case vertex) is grouped in such a way, so that if belongs to one cluster, then it cannot belong to another. On the other hand, in overlapping methods we can find an example that can belong to many clusters with different degrees of membership.
- Deterministic or stochastic: This issue is most relevant to partitional approaches designed to optimise a squared error function (a measurement of dissimilarity).
- Incremental or non-incremental: This is an issue when all examples are either not available in advance or when the set of examples is too big to be passed as a complete input to an algorithm due to memory or time constraints.

The goal of clustering is to determine the groups in a set of unlabelled data. However, measuring the quality of the final clustering depends on the criterion, defined by a user.

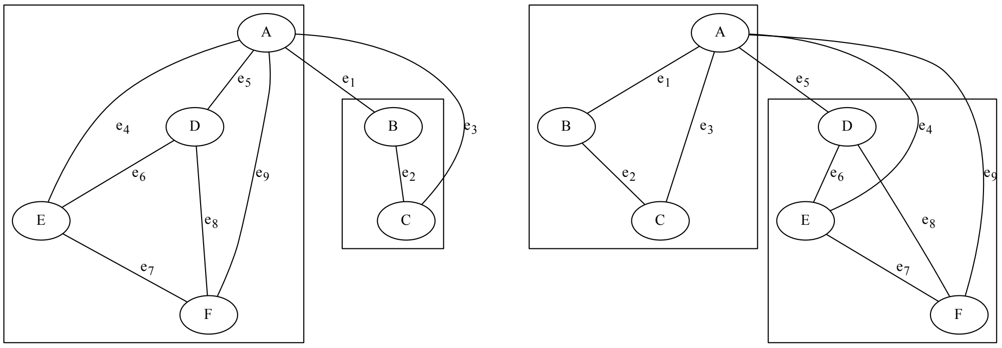
Clustering is most often done on data that are known in advance. Consider a case where an input graph G is given with all the vertices and edges. An offline clustering algorithm would in this case take the complete graph as an input and try to procure an output that is meaningful (satisfies the given criteria and number of clusters). On the other hand, consider a case where the graph is not known in advance. More specifically, vertices and edges are revealed one at a time and the algorithm has to procure an output that satisfies an objective function given upfront. This type of problem is called Online Clustering problem and is in details described in section 2.2.

We consider a case, where an input to a clustering algorithm is a graph G and we would like to define a partition of vertices into clusters such that an objective function is maximised. This type of clustering belongs to a group of correlation clustering problems and its many different variants that have been studied and introduced over the past decades [5]. The basic setup here is to cluster a collection of vertices given as input only qualitative information concerning similarity between pairs of vertices (an edge). We are not provided with any quantitate information on how different pairs of vertices are, as it is typically assumed in most other clustering formulations. Our goal is to produce a partitioning into clusters that maximises an objective function to the extend possible.

Two examples of an objective function, used in this work, are the Maximum Edge Clique Partition problem (MAX-ECP) and Minimum Edge Clique Partition problem (MIN-ECP). They are defined as the following [6]:

Definition 2.1.1. *The problem of maximum edge clique partition (Max-ECP) is defined as finding a partition of vertices V into disjoint subsets $V_1, ..., V_k$ such that for each $1 \leq i \leq k$, any two vertices in V_i share an edge and the total number of edges within the subsets $V_1, ..., V_k$ is maximised.*

In simple terms, given a general undirected graph, partition its vertices into disjoint clusters such that each cluster forms a clique and the number of edges within the clusters is maximised. For the example from Figure 3.1, we determine an optimal clustering consisting of 2 clusters, each with 1 clique. The first one consists of the vertices A, D, E and F, while the second cluster consists of nodes B and C. This clustering can be seen on Figure 2.1a.



(a) An optimal solution

(b) A feasible solution

Figure 2.1: A feasible solution (a), a sub graph of the initial graph from Figure 3.1. There are 7 edges in total inside the clusters. Clustering on Figure (b) has one edge less, thus having a profit of 6.

The definition of the minimum edge clique partition (Min-ECP) problem is similar to Max-ECP.

Definition 2.1.2. *Min-ECP aims to group a partition of vertices V into disjoint subsets $V_1, ..., V_k$ such that for each $1 \leq i \leq k$, any two vertices in V_i share an edge E_{ij}. Additionally, the total number of edges between the subsets $V_1, ..., V_k$ are minimised. [6]*

Offline correlation clustering has been studied and exhibited some results in the past. It was first studied by Ben-Dor et al. [1] motivated by some questions from computational biology and later by Shamir et al [15], proving that the problem of correlation clustering with variants of Max-ECP and Min-ECP is actually NP-hard, meaning that we can not calculate the solution in a polynomial time but in an exponential. On the other hand, Giotis and Guruswami [14] introduced a polynomial time approximation scheme (PTAS) for a case, when the number of clusters is given upfront. Such an approximation guarantees that the problem can have an approximate solution calculated in a polynomial time.

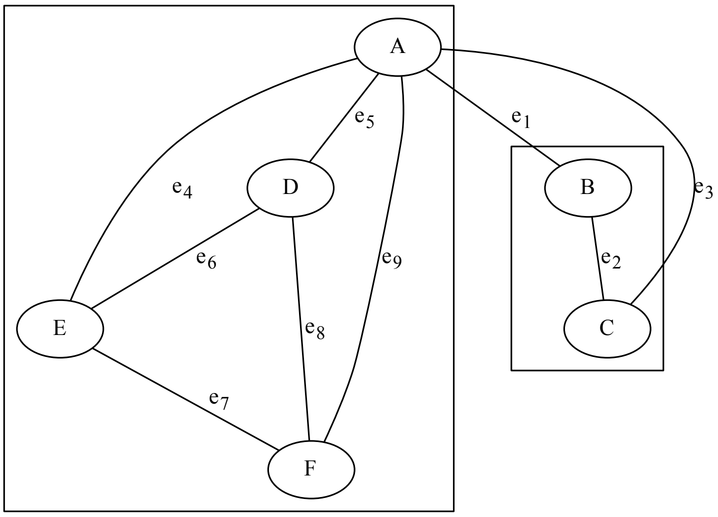


Figure 2.2: An optimal solution to Max-ECP is an optimal solution for Min-ECP. The number of edges between the clusters is minimized whenever the number of edges inside the cluster is maximized. On our example the result is 2.

Many of the algorithmic problems are identified as online. We present an online version of the correlation clustering problems with the Max-ECP and Min-ECP as the objective function in the next section.

2.2 Online Algorithms

In Online Algorithms the input is only partially known because other relevant input data will arrive in the future. An online algorithm must therefore, on every step, generate output without knowing the complete input [8]. Additionally, it would be appreciated if an output of the online algorithm does not differ too much from an output of the offline version (the one that has the complete knowledge of the input).

Formally, an online algorithm *Alg* is presented with a request sequence

$$r_1, r_2, ...r_i, r_{i+1}, r_{i+2}, ...r_n$$

by an Adversary, which provides the algorithm piecemeal information. The requests r_i, has to be served before the request r_{i+1} and it is not known in advance when the last request r_n will arrive as an input. An online algorithm Alg processes this input and if the adversary knows the strategy of the algorithm, it can construct a request sequence such that it will cause the online algorithm to behave as bad as possible. Adversary chooses data to maximise the ratio of the cost of the algorithm being studied and some optimal (e. g. offline) algorithm.
There are several applications of online algorithms [8]

- Resource management in operating systems: Paging is a classical online problem where a two-level memory system consisting of a small fast memory (L1) and a large slow memory (L2) has to be maintained. The goal is to keep actively referenced pages in fast memory without knowing which pages will be referenced in the future [16].
- Data structures: We wish to dynamically maintain this structure so that a sequence of accesses to elements can be served at low cost. Future accesses are not known. Consider a data structure such as a Most Recently Used, linear linked list or a tree [17].
- Scheduling: A sequence of jobs must be scheduled to optimise a given objective function. Jobs arrive one by one and must be scheduled immediately without knowledge of future jobs (see e.g. [18]).
- Networks: Many online problems in this area arise in the context of data transmission. The problem can be, for instance, to dynamically maintain a set of open connections between network nodes without knowing which connections are needed in the future (see e.g. [19]).
- Clustering: To maintain an optimal clustering. A vertex is introduced and has to be put in an appropriate cluster. After another vertex a computational expensive operation has to be redone due to a wrong decision a step before.

The problem of correlation clustering can also be formatted as online and it has already been studied by Mathieu et al. [3] in such version. They proved that when the objective is to minimise the disagreements (Min-ECP problem), then the natural *greedy* strategy is O(n)-competitive, and that this is optimal. When the objective is to maximise the number of agreements edges (Max-ECP), they prove that the *greedy* algorithm is $\frac{1}{2}$-competitive and that no online strategy can be better than 0.834-competitive. Additionally they purpose and prove a randomised strategy with competitive ratio slightly higher than $\frac{1}{2}$.

We introduce online clique clustering as a complementary online clustering method where each cluster is a clique. Additionally, no explicit number of clusters needs to be given upfront.

Chapter 3

Graph Theory

A graph is a mathematical structure. More specifically it is a representation of a set of objects where pairs of them, called Vertices, can be connected with Edges. A connecting edge often represents similarity whereas no edge between two vertices frequently implies that there is a difference between the two vertices [7].
Graphs can be used to model:

- Social networks
- Communications networks
- Information networks
- Software design
- Transportation networks
- Biological networks
- ...

Vertices are most commonly denoted with upper case letter, for example V_i and edges between them with lower case letters, for example e_j. In terms of mathematics, a graph G is a pair of a set V and E, denoted by $G = (V, E)$.
Vertices are most often graphically represented as a a dot, circle or any other symmetrical symbol with the label of the vertex in the middle. On the other hand, edges between two or more vertices most often appear as a line. For the purpose of this thesis we will focus on undirected graphs; however, some graphs can have edges with orientation shown with an arrow at one or both ends. These type of graphs are most commonly used to represent a flow of some quantity,

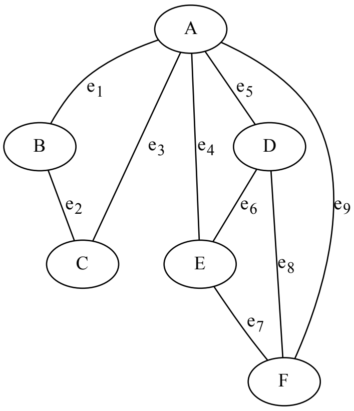


Figure 3.1: A simple undirected graph G, with a group of vertices $V = \{A, B, C, D, E, F\}$ and a set of edges $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$.

e.g. water or other fluids. The undirected graphs are, on the other hand, most commonly used to represent relations between objects, e.g. on Figure 3.1, E could be interpreted as a neighbour of A and D. Likewise, A and D are neighbours of E.

The minimum number of edges in any graph is 0, however, the maximum number of edges in an undirected graph is given in Lemma 3.0.1:

Lemma 3.0.1. *Let G be an undirected graph with n verticies. The number of edges in the graph $|E_n|$, is at most*

$$|E_n| \leq \frac{n(n-1)}{2} = \binom{n}{2} \tag{3.1}$$

Proof. This can be shown with induction over the number of vertices. A graph with two vertices can have at most 1 edge in between them. This is consistent with (3.1), therefore, the base case holds.

Let us now assume that if another vertex is introduced, the following inductive hypothesis must be true:

$$|E_{n+1}| = \frac{(n+1)*n}{2}$$

We now focus on the inductive step by introducing another vertex (which can have at most n edges to n previously existing verticies):

$$|E_{n+1}| \leq |E_n| + n$$

Using the inductive hypothesis we get

$$|E_{n+1}| \leq \frac{n(n-1)}{2} + n = \frac{n^2 - n + 2n}{2} = \frac{n(n+1)}{2}$$

which is true for any n greater or equal to 0. □

3.1 Graph Clustering

Graph clustering is the task of grouping the vertices of graph into clusters taking into consideration the number of edges inside and between the clusters. The general idea is to have many edges within each cluster and relatively few between the clusters.

One example of a criterion for graph clustering can be that we would be interested in finding clusters that contain only vertices with the same number of edges (identical degree of the vertex). We name this problem MaxDeg and additionally to the proposed criterium, the degree of the vertex ($deg(v)$) should be maximised. If all of the vertices have the same degree (identical number of edges), we say that the graph is regular.

Graphically we can see one feasible solution to MaxDeg problem of graph from Figure 3.1 on Figure 3.2.

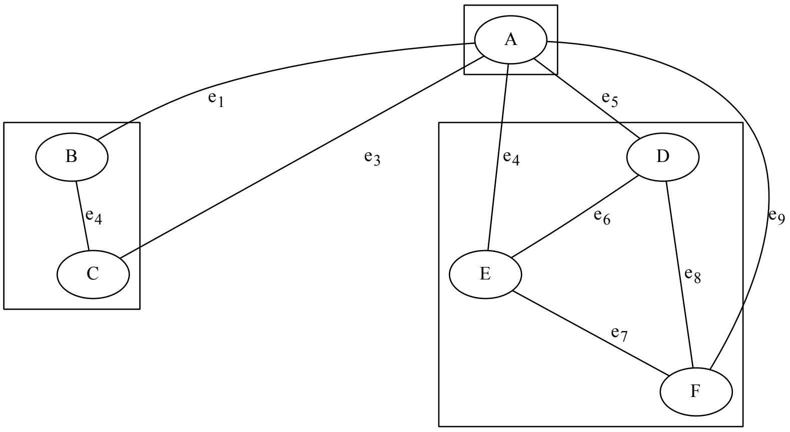


Figure 3.2: A feasible clustering of the initial graph from Figure 3.1. The first cluster contains only vertex A with a degree of five, the second cluster consists of vertices D,E and F which all have a degree of 3. The last cluster contains vertices B and C with a degree of 2. Each cluster satisfies the criterium of Max-ECP and is at the same time also a regular graph on its own.

3.2 Clique Graph

Definition 3.2.1. *Graph G is a clique if there exists an edge $e_{i,j}$ for every pair of vertices $\{V_i, V_j\} \in G$, where $i \neq j$.*

An example of such a graph can be seen on Figure 3.3. A clique graph represents a possible clustering of the vertices and due to its definition the number of edges inside the clusters is limited by (3.1).

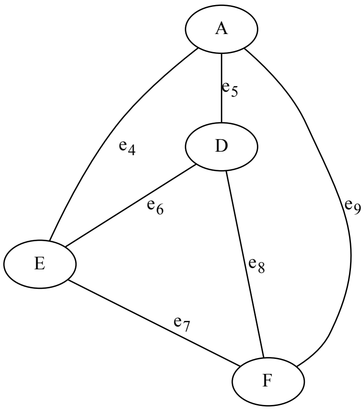


Figure 3.3: A feasible Clique, a sub graph of the initial graph from Figure 3.1. There is an edge in-between all of the vertices.

3.3 Profit function

In order to measure how successful an operation on a graph is we introduce a function called profit. It is problem specific and measures the gain from an operation. For Maximum Edge Clique Partition (Max-ECP) problem we define profit as the number of edges inside a cluster and for Minimum Edge Clique Partition (Min-ECP) problem the number of edges between clusters respectively. Similarly we define the cost to be the consequence of the operation resulting in the case of Max-ECP to be the number of edges between clusters and in the case of Min-ECP to be the number of edges inside clusters. The sum of profit and cost equals to the number of edges in the graph. The edges that count in profit are called agreements and those that count as a cost, disagreements [3].

3.3.1 Profit definition for Max-ECP and Min-ECP

Let C_i be the current clustering on i vertices. Moreover, let C_{i+1} be a clustering on $(i+1)$ vertices after a new vertex is introduced.

Definition 3.3.1. *We define the profit function* profit(C) *to be the difference of edges that are inside a cluster (between clusters) for the MAX-ECP (MIN-ECP) problem in respect to the clusterings C_i and C_{i+1}.*

3.4 Competitive Ratio

The idea of competitiveness is to compare the output generated by an online algorithm to the output produced by an optimal algorithm, often denoted with OPT. The later is an optimal offline algorithm that knows the entire input data in advance and can compute an optimal output.

Definition 3.4.1. *We define competitive ratio Comp as the ratio between profits of an online version of an algorithm (Alg) and an offline version of the optimal algorithm (OPT).*

$$Comp = \frac{\text{profit(Alg)}}{\text{profit(OPT)}}$$

The better an online algorithm approximates the optimal solution, the more competitive this algorithm is. If we have two algorithms Alg_1 and Alg_2 and we compute that Alg_1 has a competitive ratio of $Comp_1$ which is greater than $Comp_2$ (competitive ratio of Alg_2), we say, that algorithm Alg_1 is more competitive than Alg_2. In case of the Max-ECP problem defined in section 2.1.1, the outcome of Alg_1 would for example produce a clustering on a graph G with more edges inside the clusters than Alg_2.

3.5 Complexity of Max-ECP and Min-ECP

In [5], Bansal et al. show that both the minimisation (minimising the number of disagreement edges) and the maximisation (maximising the number of agreement edges) versions are in fact NP-hard. This means that we can verify in polynomial time if a given solution to an ECP problem is correct, however, do not know it in polynomial time. From the point of view of approximation the maximisation and minimisation versions of ECP problems differ.

In the case of maximising agreement edges (Max-ECP) this problem actually admits a polynomial-time approximation scheme, meaning that an algorithm for finding an approximate solution needs polynomial number of steps to complete.

In the case of minimising disagreements (Min-ECP) it is APX-hard, meaning that there exists a constant c such that it is NP-hard to find an approximation algorithm with approximation ratio better than c. Problems in this class have efficient algorithms that can find an answer within some fixed percentage of the optimal answer. Several efficient constant factor approximation algorithms are proposed in the literature when minimising disagreements (Min-ECP problem) [5].

Chapter 4

Online Max-ECP Clustering

In online clique clustering an input graph $G = (V, E)$ is given with a goal of finding a clustering that satisfies the Max-ECP or Min-ECP problems. Vertices are given in a serial fashion, meaning that they are not known in advance. Additionally, the number of vertices is not known either.

Upon the arrival of a new vertex, an online algorithm has to update the current clustering in on of the following ways:

- vertex is being added to a pre-existing cluster if it satisfy the criterium
- a new singleton cluster is created (a cluster with only one vertex) if it does not satisfy the criterium
- some pre-existing clusters can be merged

4.1 The Greedy Strategy

The *greedy* strategy for Max-ECP clustering merges each input vertex with the largest current cluster that maintains the clique property. If no such merging is possible the vertex is placed into its own (also called singleton) cluster. *Greedy* strategies are natural first attempts used to solve online problems and can be shown to behave well for some problems, however, the *greedy* strategy can be far from optimal for Max-ECP clustering [9].

To demonstrate this let us look at an example how *greedy* would perform on our initial graph from Figure 3.1 on page 4.

The input of the vertices has to be serialised; $V = \{A, B, C, D, E, F\}$. The adversary first introduces vertex A to the algorithm *greedy*.

- Vertex A is clustered into it's own (singleton) cluster as there are no other clusters yet existing, Figure 4.1a.
- Next in Line is vertex B with an edge $e_{A,B}$. Due to the shared edge with A, *greedy* merges the Vertex B together with A.
- Vertex C shares an edge with both A and B, therefore becoming a part of the same cluster as well, Figure 4.1b.
- Next in the queue is $D$with an edge $e_{D,A}$. It shares an edge with A, however, not with B and C. The split operation is not allowed so the only possible outcome here is to put D into a singleton cluster, Figure 4.1c.
- Vertices E and F share an edge with D and each other so they become a part of the cluster 2 together with D, Figure 4.1d.

Greedy stops as there are no more vertices given by the adversary. The number of edges inside clusters is being counted and in the example above the profit equals to 6 (in both cliques there are 3 edges). *Greedy* strategy is far from optimal and is no better than $\frac{2}{|V|-2}$-competitive, as visible from theorem 4.1.1.

Theorem 4.1.1. *The* greedy *strategy for Max-ECP clustering is no better than* $\frac{2}{(n-2)}$*-competitive.*

Proof. Consider an adversary that provides input to the strategy to make it behave as badly as possible.

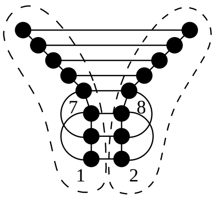
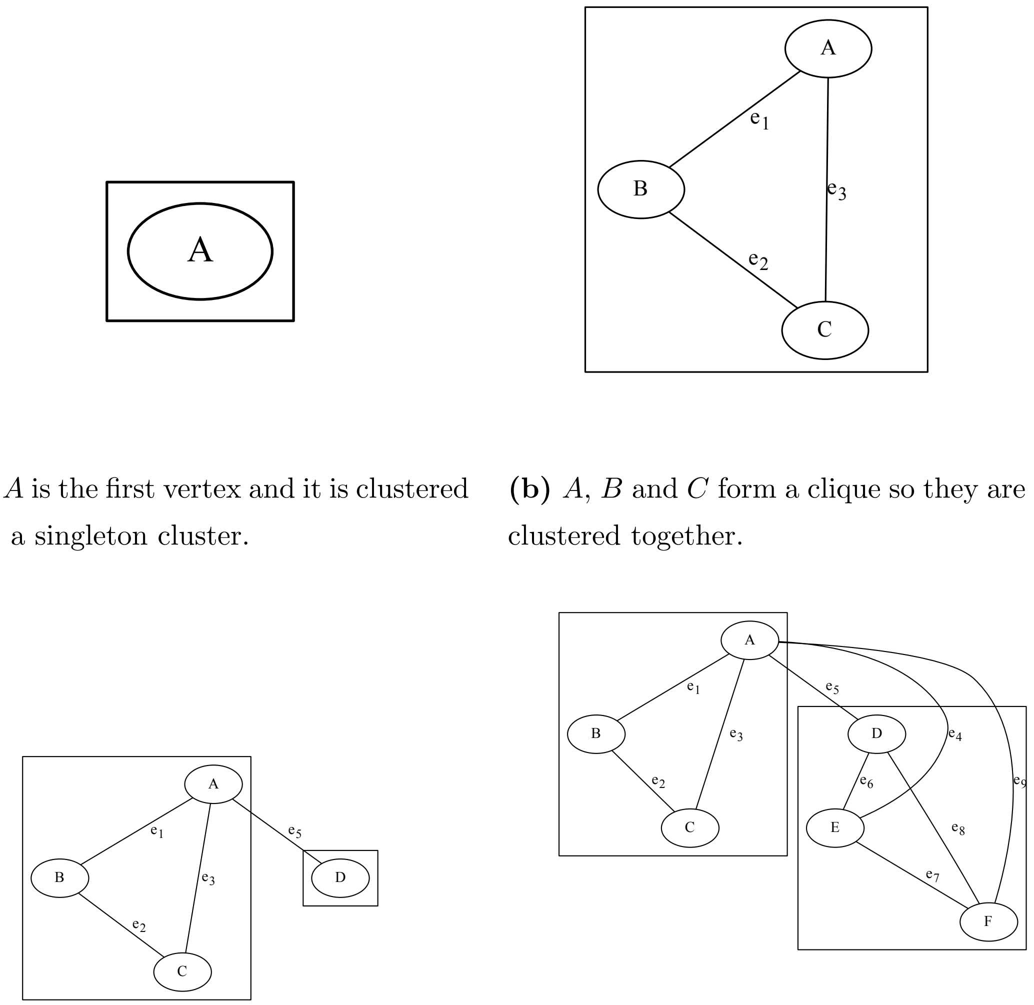


Figure 4.2: Illustrating the proof of Theorem 4.1.1.

Our adversary gives *greedy* $n = 2k$ vertices in order from 1 to $2k$. Each odd numbered vertex is connected to its even successor, each odd numbered vertex is



(a) A is the first vertex and it is clustered into a singleton cluster.

(b) A, B and C form a clique so they are clustered together.

(c) Since vertex D does not have an edge to B and C, it has to be clustered separately.

(d) D, E and F form a clique so they are clustered together. A is a member of the previous cluster.

Figure 4.1: Greedy procured 2 clusters that equally contribute both 3 edges, summing up the profit to 6. A careful observation can show that OPT procures a profit of 7.

also connected to every other odd numbered vertex before it, and similarly, each even numbered vertex is also connected to every even numbered vertex before it; see Figure 4.2.

The *greedy* strategy clusters the vertices as odd/even pairs, giving the clustering GDY_n having profit $\text{profit}(GDY_{\text{n}}) = k$. An optimal strategy clusters the odd vertices in one clique of size k and the even vertices in another clique also of size k. The profit for the optimal solution is $\text{profit}(OPT_{\text{n}}) = k(k-1)$. Hence, the worst case ratio between *greedy* and an optimum solution is at most $1/(k-1) = 1/(n/2-1)$ so the competitive ratio is at most $2/(n-2)$.

□

4.2 Lower Bound

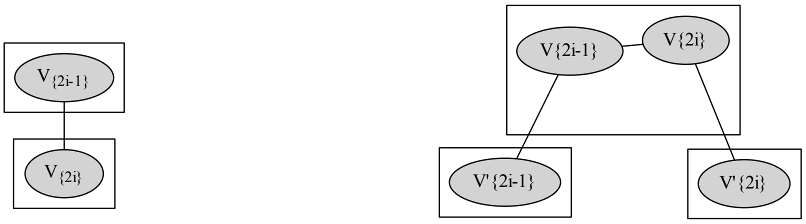
Theorem 4.2.1. *Any deterministic strategy for Max-ECP clustering is at most* $\frac{1}{2}$ *- competitive.*

Proof. Let the adversary provide $2k$ vertices, where every odd numbered vertex is connected to its even numbered vertex, V_1 to V_2, V_3 to V_4, etc. The game now continues in stages with the strategy constructing clusters followed by adding edges. In each stage the adversary looks at the clusters constructed; these are either singletons or pairs $\{V_{2i-1}, V_{2i}\}$. For each newly constructed pair cluster, the adversary adds two new vertices, V'_{2i-1} connected to V_{2i-1}, and, V'_{2i} connected to V_{2i}. The adversary stops providing new vertices as soon as the strategy can not produce a pair in a cluster at any stage.

Assume that the strategy at the end of the stages has constructed k' pair clusters, $k' \leq k$, thus giving a profit of k'. Note that the strategy can never produce the pairs $\{V_{2i-1}, V'_{2i-1}\}$ or $\{V_{2i}, V'_{2i}\}$ since these are revealed only if the pair $\{V_{2i-1}, V_{2i}\}$ is produced. The optimal solution in this case has profit $k + k'$ since this solution produces $2k'$ pair clusters $\{V_{2i-1}, V'_{2i-1}\}$ or $\{V_{2i}, V_{2i}\}$, where the strategy produces $\{V_{2i-1}, V_{2i}\}$, in addition to $k - k'$ pairs $\{V_{2i-1}, V_{2i}\}$, where the strategy produces singleton clusters. Hence, the competitive ratio is

$$\frac{k'}{k+k'} \leq \frac{1}{2}, \qquad \text{for } 0 \leq k' \leq k, \tag{4.1}$$

□



(a) Every odd numbered vertex is connected to its even successor.

(b) Two new vertices (marked with ') are introduced.

Figure 4.3: First pair of vertices (a) is introduced. Due to an existing edge they will be clustered together. Additionally, a pair of vertices is introduced on the next step (b) which can only be clustered into singleton clusters.

Based on (4.1) we can conclude that any new improved strategy introduced cannot have a better competitive ratio than $\frac{1}{2}$. This reveals the possibility of an Algorithm that can keep a constant competitive ratio, independent of the number of nodes. One such algorithm is presented in section 4.3.

4.3 A Lazy Strategy

We present a new strategy for Max-ECP clustering and prove that it has constant competitive ratio. We divide its execution between two cases:

- If the optimum solution OPT_n (an offline optimum clustering performed on n vertices) does not have any clusters of size larger than three, the strategy follows the *greedy* strategy
- Otherwise, the strategy places arriving vertices in singleton clusters until the profit ratio between the current solution S'_n and the offline optimum solution OPT_n (of the n currently known vertices) goes below a threshold value. When this happens the strategy computes the *relative optimum* (optimum clustering on the vertices and edges revealed) solution given the current clustering

The strategy is given in pseudocode on Figure 4.4 bellow.

```
Strategy  Lazy
    /* Maintain $S_n$ with profit $\mathrm{profit(S_n)}$ and let $c$ be a constant */
1       $n = 1$
2       while new vertex $v_n$ arrives do
2.1         $S'_n := S_{n-1} + \mathit{singleton}(v_n)$
2.2         Compute $OPT_n$
2.3         if the largest cluster in $OPT_n$ has size $\geq 4$ then
2.3.1           if $\mathrm{profit(OPT_n)} > c \cdot \mathrm{profit(S'_n)}$ then
2.3.1.1             Compute the relative optimum of $S'_n$, $\widehat{OPT}(S'_n)$
2.3.1.2             Construct $S_n$ from $\widehat{OPT}(S'_n)$
                else
2.3.1.3             $S_n := S'_n$
                endif
            else
2.3.2           Construct $S_n$ using the greedy strategy
            endif
2.4         $n := n + 1$
        endwhile
End  Lazy
```

Figure 4.4: The Lazy strategy

Given a clustering S, the *relative optimum*, $\widehat{OPT}(S)$, is defined as follows: construct a graph G_S such that, for every cluster in S there is a vertex in G_S and two vertices in G_S are connected by an edge, if every pair of vertices in the two underlying clusters are connected. $\widehat{OPT}(S)$ is now the offline optimal clustering in G_S.

Given the current clustering, S'_n, the new clustering, S_n, is easily generated by constructing a cluster in S_n for each cluster in $\widehat{OPT}(S'_n)$ by merging the corresponding clusters in S_n.

The following definition follows directly from the construction of the strategy.

Theorem 4.3.1. *There is a constant c such that the Lazy strategy is* $1/c$*-competitive for online Max-ECP clustering.*

We prove later that Theorem 4.3.1 holds for $c = (154 + 16\sqrt{61})/9$.

4.3.1 Simulating Greedy and Lazy

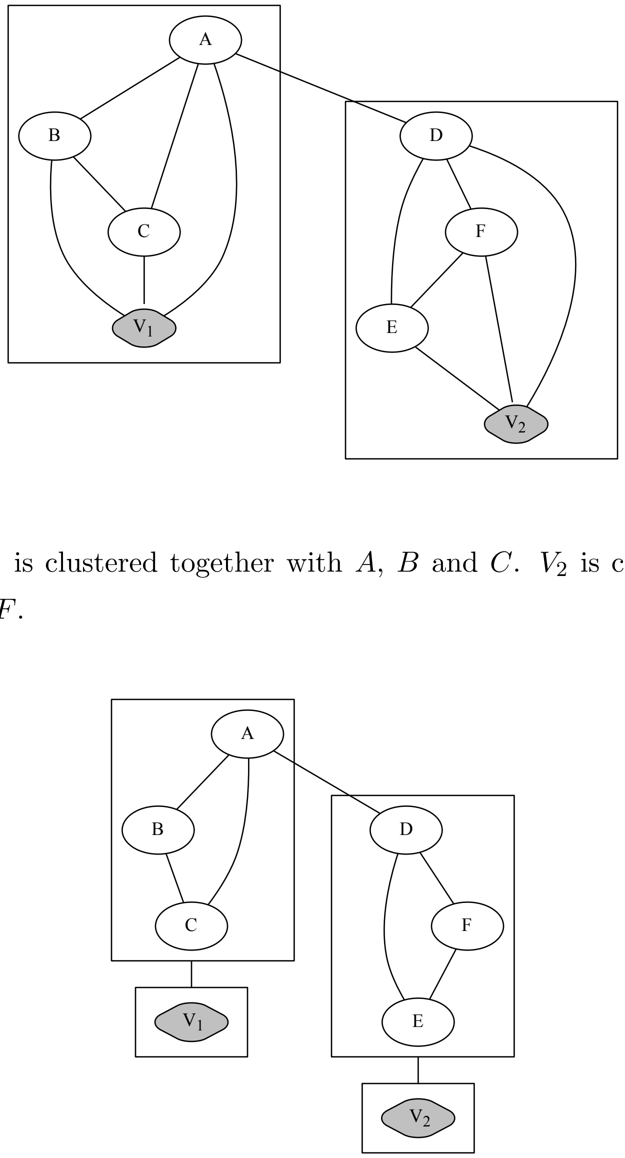
In order to understand the benefits provided by the strategy, we first show how *greedy* would have performed, using our initial graph with additional vertices. We then compare the clustering induced by *greedy* with the clustering done by the Lazy strategy.

The scenario is the following: Adversary gives us the vertices from A to F with edges, same as in Figure 3.1. *Greedy* will by nature create 2 clusters, the first one containing vertices A, B and C, while the second one will consist of D, E and F. Graphically this is identical to Figure 2.1b. Counting the *profit* we see that each of the cliques yields 3, summing up to 6. The later does not deviate from the before mentioned optimum of the current scenario being 7.

The lazy strategy on the other hand follows the *greedy*, yielding the *profit* 7 with constructing the identical cliques. This happens because the condition in Step 2.3 is not fulfilled.

Now let us assume that the adversary yields vertices V_1 and V_2, where V_1 has an edge to all the vertices in one of the cliques (for example, to A, B and C) and V_2 an edge to all the vertices in the other clique (D, E and F) respectively. *Greedy* due to its nature clusters V_1 together with A, B and C increasing the cluster in size. Respectively, V_2 is clustered together with D, E and F. Graphically this can be seen on Figure 4.5a.

Lazy on the other hand clusters the newly arrived vertices into singleton clusters. This is the case due to condition in Step 2.3 being fulfilled (Figure 4.5b).



(a) Greedy; V_1 is clustered together with A, B and C. V_2 is clustered together with D,E and F.

(b) Lazy; V_1 and V_2 form both a singleton cluster.

Figure 4.5: Greedy merged V_1 and V_2 into existing clusters, while Lazy produces 2 new singleton clusters.

The simulation continues by adversary giving us new vertices, labeled from V_3 to V_{20}, having each odd ancestor (vertex with an index greater than the current) vertex connected to every existing odd vertex and each even ancestor connected to every existing even vertex.

Note that *greedy* has previously merged V_1 and V_2 to an existing cluster and that V_3 and V_4 do not have an edge to all the vertices in the cluster, meaning that immediately after introducing V_3 and V_4, *greedy* creates two new singleton clusters. Every odd vertex is clustered together with V_3 and every even with V_4. The end product is a Graph, consisting of 4 clusters as shown on figure 4.6.

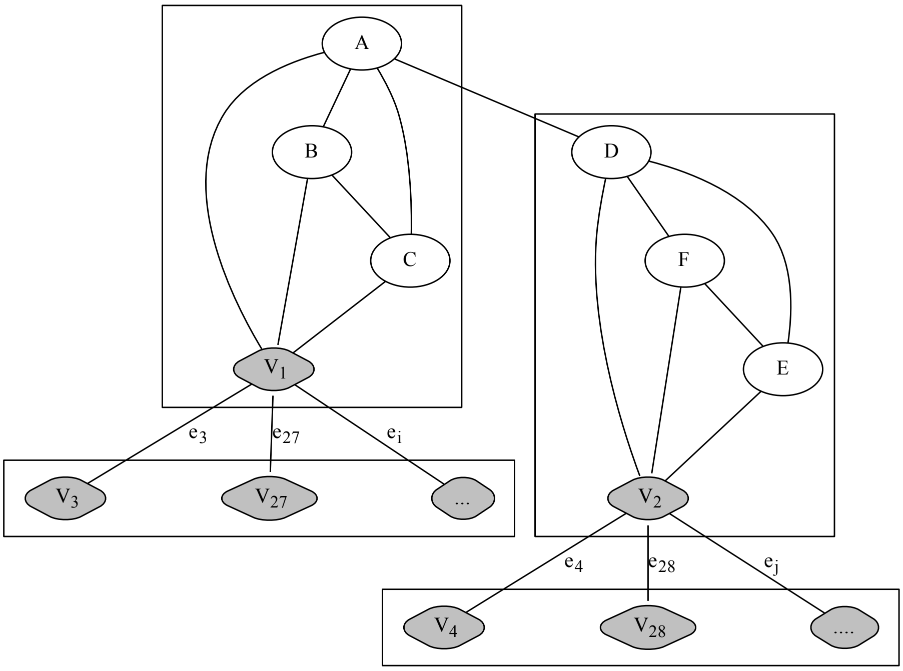


Figure 4.6: Greedy ends with 4 clusters. Mind that edges from e_3 up to e_{28} are not inside any of the clusters. They are disagreements and therefore do not count to the profit.

We calculate the profit of *greedy* by counting the number of edges inside the clusters. The first two clusters each contribute 6 while the bigger clusters contain both 13 vertices. Using the formula 3.1 we see that the profit of the *greedy* strategy is 168.

$$\text{profit}(\text{greedy}_{28}) = \binom{4}{2} + \binom{4}{2} + \binom{13}{2} + \binom{13}{2} = 168$$

Lazy on the other hand expects the new vertices with 2 clusters of size 3 and 2 singleton clusters with V_1 in the first one and V_2 in the second one respectively. As per the scenario, every new odd vertex has an edge to every existing odd vertex. However, due to the condition in Step 2.3.1 not being fulfilled, vertices from V_2 up to V_{27} can not be clustered together in any possible way. They are put into a singleton and the simulation continues with the existing clustering and, in each step, one new cluster as per Step 2.3.1.3.

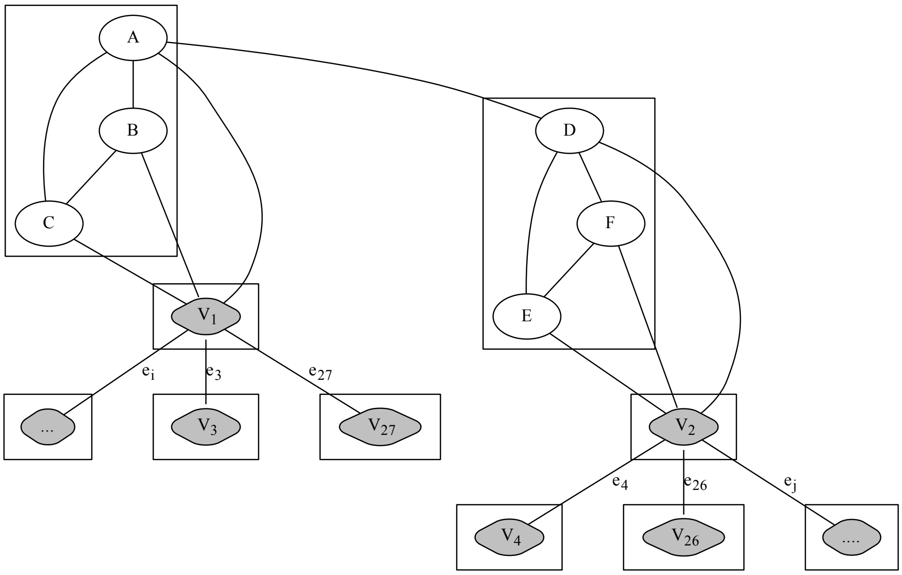


Figure 4.7: Newly arrived vertices from V_1 until V_{27} are being clustered separately into singleton clusters. The profit of the offline solution has to reach a threshold value for the actual clustering to happen.

This continues until the condition Step 2.3.1 from Lazy is fulfilled. The profit of the strategy S'_n equals to 6 (contributing each of the initial clusters 3). To follow with the strategy, we assign c to 30.99 (the selection is determined by a later introduced Lemma 4.3.3) A simple calculation using (4.2) shows that the first time when this condition can be fulfilled is when the vertex V_{28} is introduced (making the profit of the offline strategy OPT_n greater than the right side of the equation).

$$\mathrm{profit}(\mathrm{OPT_n}) > 30.99 \cdot 6 \tag{4.2}$$

At this point of time, a new optimal clustering is being calculated and as per the scenario, Lazy executes a merge operation, joining odd vertices together in a cluster (including V_1) and even vertices together (including V_2) respectively. This is shown on Figure 4.8.

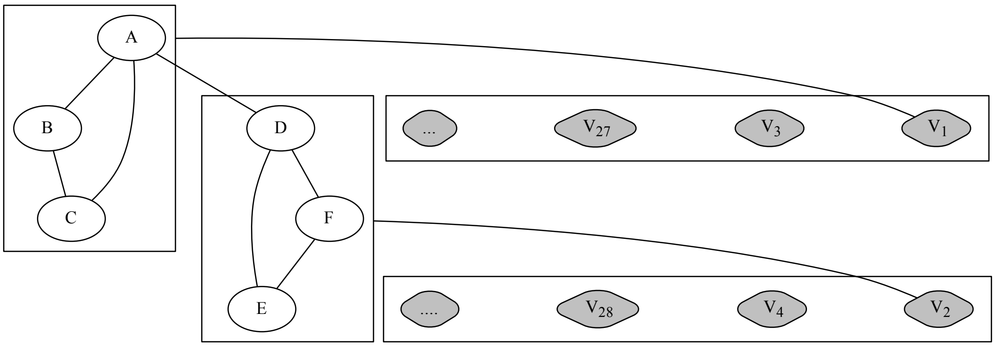


Figure 4.8: The newly arrived vertex V_{28} causes the Lazy strategy to start executing merge operations, forming an instant optimal clustering. The end result are 4 clusters, however two clusters have a significantly higher amount

Likewise, we count the number of edges inside the clusters. In this case, Lazy produced 4 clusters with symmetrical *profit*s. The first two contribute 3 each, while the second two contain each 14 vertices. Using the (3.1) we see that the profit of the Lazy strategy is 188.

$$\mathrm{profit(S_{28})} = \binom{3}{2} + \binom{3}{2} + \binom{14}{2} + \binom{14}{2} = 188$$

4.3.2 Comptetitivnes of the strategy

We begin the proof by giving a relationship between the profits of the two clusterings OPT_{n-1} and OPT_n.

Lemma 4.3.1. *Let $c_{\max}$ be the largest cluster in OPT_{n-1}, with size k. If this is the case, for all $n > 2$,* $\mathrm{profit}(OPT_\mathrm{n}) \leq \mathrm{profit}(OPT_\mathrm{n-1}) \cdot \frac{(k+1)}{(k-1)}$.

Proof. The maximum increase occurs if the next vertex given by the adversary v_n can be clustered together with the vertices in $c_{\max}$. The increase in profit in this cluster goes from $\binom{k}{2}$ to $\binom{k+1}{2}$. The maximum increase for the whole clustering happens when $c_{\max}$ is the only non-singleton cluster in OPT_{n-1}, giving a ratio of $\binom{k+1}{2}/\binom{k}{2} = \frac{(k+1)}{(k-1)}$. □

Let G be an undirected graph and let G_A and G_B be two subgraphs produced by some partitioning of the vertices in G. Let C be a clustering on G and let A and B be the clusterings induced by C on G_A and G_B respectively.

Lemma 4.3.2. *If* $\mathrm{profit(A)} > 0$ *and* $\frac{\mathrm{profit(C)}}{\mathrm{profit(A)}} = z > 1$*, then* $\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} \geq r(z)$ *where*

$$r(z) = 1 - \frac{\sqrt{1+8z} - 2}{z}.$$

Proof. The proof is by a two level induction on the number of clusters in C. We can assume that the clusters $c_1, \ldots, c_m$ in C are sorted on increasing number of vertices in a_i, where a_i is the cluster in A induced by the cluster c_i in C. Similarly we denote by b_i the cluster in B induced by the cluster c_i in C.

A cluster c_i is a *null cluster*, if the induced cluster a_i in A has profit of 0 ($\mathrm{profit(a_i)} = 0$). This occurs if a_i does not contain any vertices or contains exactly one, being a singleton cluster.

The first step in every induction is to prove the base case. Here we assume that cluster C contains exactly one non-null cluster, i.e., $c_1, \ldots, c_{j-1}$ are clusters such that $\mathrm{profit(a_i)} = 0$, for $1 \leq i < j$ and c_j is the first cluster where $\mathrm{profit(a_j)} > 0$. With other words, a_j contains at least 2 nodes connected by 1 edge. Assume that $\mathrm{profit(c_j)/profit(a_j)} = z''$ and that $|a_j| = l$, $|b_j| = l'$ and $|c_j| = l + l'$.

We prove the base case of the induction also using induction and set for this base case j to 1. In this case,

$$z = \mathrm{profit(C)/profit(A)} = \mathrm{profit(c_1)/profit(a_1)} = z''$$

and we get that by using the inductive hypothesis ($\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} \geq r(z)$), that

$$\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} = \frac{\mathrm{profit(b_1)}}{\mathrm{profit(c_1)}} = 1 - \frac{\sqrt{1 + 4zl(l-1)} - l}{z(l-1)} \geq r(z),$$

since the ratio is increasing in l.

For the inductive case of the base case, we assume the result holds for $j-2 \geq 0$ null clusters and one non-null cluster and prove it for $j-1$ null clusters and one non-null cluster. Let $\{c_2, \ldots, c_j\}$ be denoted by C' and let A' and B' be the induced clusterings of C' in G_A and G_B. We set $\mathrm{profit(C')/profit(A')} = z'$ and have when we add null cluster c_1 to the clustering that

$$z = \frac{\mathrm{profit(C)}}{\mathrm{profit(A)}} = \frac{\mathrm{profit(C') + profit(c_1)}}{\mathrm{profit(a_j)}} = z' + \frac{\mathrm{profit(c_1)}}{\mathrm{profit(a_j)}},$$

giving us that $z' \leq z$ and

$$\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} = \frac{\mathrm{profit(b_1) + profit(B')}}{\mathrm{profit(C)}} \geq \frac{\mathrm{profit(c_1)} - |c_1| + 1 + \mathrm{profit(B')}}{\mathrm{profit(c_1) + profit(C')}}.$$

By the induction hypothesis we have that $\mathrm{profit(B')} \geq r(z') \cdot \mathrm{profit(C')}$ giving us after proper substitutions that

$$\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} \geq 1 - \frac{z'\sqrt{1+8z} - 2z'}{z^2} - \frac{\sqrt{1+8z-8z'} - 1}{2z}.$$

The right side is a decreasing function of z' (the derivative is negative) so increasing z' to z yields $\mathrm{profit(B)/profit(C)} \geq r(z)$. This proves the base case when C has zero or more null clusters and exactly one non-null cluster.

For the general induction step we assume the formula holds for $m-1$ clusters and we prove it for m clusters. Let

$$z = \frac{\mathrm{profit(C)}}{\mathrm{profit(A)}},$$

$$C' = \{c_1, \dots, c_{m-1}\}, C = C' \cup \{c_m\},$$

$$z' = \frac{\mathrm{profit(C')}}{\mathrm{profit(A')}}$$

and

$$z'' = \frac{\mathrm{profit(c_m)}}{\mathrm{profit(a_m)}}$$

.

By the induction hypothesis we have that

$$\begin{aligned}\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} &\geq \frac{r(z')\cdot\mathrm{profit(C')} + r(z'')\cdot\mathrm{profit(c_n)}}{\mathrm{profit(C')} + \mathrm{profit(c_n)}} \\ &= r(z')\frac{z'(z-z'')}{z(z'-z'')} + r(z'')\frac{z''(z'-z)}{z(z'-z'')}\end{aligned}$$

The last expression decreases as z'' tends towards z, again giving us $\frac{\mathrm{profit(B)}}{\mathrm{profit(C)}} \geq r(z)$, thus proving our result. □

Lemma 4.3.3. *If, for a certain value of n, the selection in Step 2.3.1 yields true in the lazy strategy, then* $\mathrm{profit}(OPT_\mathrm{n}) \leq a \cdot \mathrm{profit(S_n)}$ *where $a < c$ is some constant.*

Proof. When the largest clusters in OPT_n has size at most three, we have from the proof of the *greedy* bound 4.1 that *greedy* has competitive ratio 1/4, and lazy will do at least as well in this case, since it follows the *greedy* strategy. So, we can assume that the largest cluster in OPT_n has size at least four. This also means that the size of the largest cluster in OPT_{n-1} is at least three.

We make a proof by induction on n, the number of steps in the algorithm. The base cases when $n = 1$, 2 and 3 follow immediately, since lazy (and *greedy*) computes optimal solutions in these cases, so $a \leq 4$ can be chosen as the constant, since the competitive ratio is $1/a \geq 1/4$.

Assume for the inductive case that Step 2.3.1 yields true at the n-th iteration and assume further that the previous time it happened it was in iteration k (or that the strategy followed *greedy* in this step). By the induction hypothesis we know that $\mathrm{profit}(OPT_\mathrm{k}) \leq a \cdot \mathrm{profit(S_k)}$ for some constant $a < c$. Let OPT'_k be the clustering obtained from OPT_n induced by the vertices $v_1 \dots v_k$. It is obvious that $\mathrm{profit}(OPT'_\mathrm{k}) \leq \mathrm{profit}(OPT_\mathrm{k})$. Let E_{kn} be the set of edges between vertices inside clusters of OPT_n that have both endpoints among the vertices $v_{k+1} \dots v_n$. Similarly, we define E'_{kn} to be the set of edges inside clusters that have one end point among the vertices $v_1 \dots v_k$ and the other among $v_{k+1} \dots v_n$. We now have that

$$\text{profit}(OPT'_{\text{k}}) + |E'_{kn}| + |E_{kn}| = \text{profit}(OPT_{\text{n}}).$$

Let S'_n be the clustering solution in iteration n just before the strategy reaches Step 2.3.1, i.e., when vertex v_n is put in a singleton cluster. This gives us, since $\text{profit}(\text{S}'_{\text{n}}) = \text{profit}(\text{S}_{\text{k}})$,

$$\text{profit}(OPT_{\text{n}}) \quad > \quad c \cdot \text{profit}(\text{S}'_{\text{n}}) = c \cdot \text{profit}(\text{S}_{\text{k}}) \geq \frac{c}{a} \cdot \text{profit}(OPT'_{\text{k}})$$

Since $\frac{\text{profit}(OPT_{\text{n}})}{\text{profit}(OPT'_{\text{k}})} \geq \frac{c}{a}$, by Lemma 4.3.2 the ratio $\frac{|E_{kn}|}{\text{profit}(OPT_{\text{n}})} \geq r(\frac{c}{a})$.

Note that E_{kn} forms a clustering of vertices $v_{k+1}, \dots, v_n$ that is independent of how vertices $v_1, \dots, v_k$ are clustered. Therefore, when a new cluster S_n is recomputed in Step 2.3.1.1, it includes at least as many edges as both S_k and E_{kn} together. Furthermore, $\text{profit}(\text{S}_{\text{n}-1}) = \text{profit}(\text{S}_{\text{k}})$ and $\text{profit}(OPT_{\text{n}-1}) \leq c \cdot \text{profit}(\text{S}_{\text{n}-1})$, since otherwise Step 2.3.1.1 would have been done already in the previous iteration. We have that

$$\begin{aligned}
\text{profit}(\text{S}_{\text{n}}) &\geq \text{profit}(\text{S}_{\text{k}}) + \text{profit}(OPT_{\text{kn}}) \geq \text{profit}(\text{S}_{\text{k}}) + |E_{kn}| \\
&= \text{profit}(\text{S}_{\text{n}-1}) + |E_{kn}| \\
&\geq \frac{\text{profit}(OPT_{\text{n}-1})}{c} + |E_{kn}| \geq \frac{\text{profit}(OPT_{\text{n}})}{2c} + |E_{kn}| \\
&\geq \frac{\text{profit}(OPT_{\text{n}})}{2c} + r(\frac{c}{a}) \cdot \text{profit}(OPT_{\text{n}}).
\end{aligned}$$

The second to last inequality follows from Lemma 4.3.1, since the largest cluster in OPT_{n-1} must have size 3, and the last inequality was given above.

We must guarantee that

$$\frac{\text{profit}(OPT_{\text{n}})}{2c} + r(\frac{c}{a}) \cdot \text{profit}(OPT_{\text{n}}) \geq \frac{\text{profit}(OPT_{\text{n}})}{a}$$

to prove the lemma, which is equivalent to finding constants $a \leq 4$ and c as small as possible so that $1/(2c) + r(\frac{c}{a}) \geq 1/a$. The expression holds for $a = 4$ and in equality for $c = (154 + 16\sqrt{61})/9 \approx 30.9960$. □

From Theorem 4.3.1 it follows that if $c = (154 + 16\sqrt{61})/9$, the competitive ratio for the lazy strategy equals to $9/(154 + 16\sqrt{61}) \approx 0.032262$.

Chapter 5

Online Min-ECP Clustering

5.1 A Lower Bound

In this section, we prove that there cannot exist a deterministic strategy to solve the MIN-ECP problem and would be better than $\frac{n^{1-\epsilon}}{2}$-competitive. Additionally, we prove that the *greedy* strategy for MIN-ECP yields a competitive ratio of $n-2$.

Theorem 5.1.1. *Any deterministic strategy for Min-ECP clustering is no better than $\frac{n^{1-\epsilon}}{2}$-competitive, for every $\epsilon > 0$.*

Proof. An adversary provides the following vertices of an input graph in a sequence. First, one $(k-1)$-clique, followed by one additional vertex connected to one of the previously given vertices such that a lollipop graph is formed (see Figure 5.1). We now consider different possibilities for clustering the k vertices.

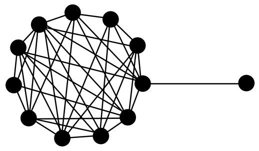


Figure 5.1: A lollipop graph with a $(k-1)$-clique and a vertex connected by a single edge.

First, let us assume that the strategy has clustered the input in such a way that the $(k-1)$-clique is not clustered as a cluster. Then this clustering has at least $k-2$ disagreements. An optimal clustering contains only one disagreement, between the $(k-1)$-clique and a singleton cluster containing the vertex outside the clique. Hence, the competitive ratio in this case is at least $(k-2)/1 = n-2$ since the number of vertices is $n = k$.

Assume next that the strategy has clustered the input as one $(k-1)$-clique and one singleton cluster. In this case, the adversary provides $k-1$ independent cliques of size m, where each of the vertices in an m-clique is connected to one particular vertex of the original $(k-1)$-clique as visible on Figure 5.2.

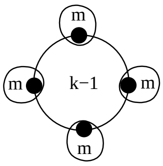


Figure 5.2: Each of the $(k-1)$ vertices in the central clique is connected with m edges to an m-clique.

The strategy can at best cluster the $k-1$ m-cliques as clusters, thus generating $m(k-1)$ disagreements. An optimal solution will, for m sufficiently large, cluster the vertices in the original $(k-1)$-clique in each of the new cliques, generating a solution of $k-1$ $(m+1)$-cliques. This solution has $\binom{k-1}{2}$ disagreements. If we set $m = (k-2)^t/4$, where k is chosen so that m is an integer and t is some sufficiently large integer, then the competitive ratio becomes

$$\frac{m(k-1)}{\binom{k-1}{2}} = \frac{(k-2)^{t-1}}{2} \geq \frac{1}{2}\left(n^{1/(t+1)}\right)^{t-1} \geq \frac{n^{1-\frac{2}{t+1}}}{2} = \frac{n^{1-\epsilon}}{2},$$

for all $\epsilon > 0$, since the number of vertices in the input is

$$n = m(k-1) + k = \frac{(k-2)^t}{4}(k-1) + k \leq (k-2)^{t+1},$$

proving the lower bound. □

5.2 The Greedy Strategy for Min-ECP

Greedy strategy for Min-ECP problem clusters the arrived vertex V_i in such a way that the number of edges between clusters is minimised. V_i is clustered together with the nodes with which it shares the most edges with. If V_i does not have an edge to all the nodes in any existing cluster, it is put into a singleton cluster.

Theorem 5.2.1. *The greedy strategy for Min-ECP clustering is no better than* $\frac{(n-2)}{2}$*-competitive.*

Proof. We let an adversary generate the same input sequence of $n = 2k$ vertex pairs as in the proof of Theorem 4.2.1. *Greedy* generates $2\binom{k}{2}$ disagreement edges whereas the optimum solution has only k disagreement edges. The competitive ratio is, by using Definition 3.4.1,

$$\frac{2\binom{k}{2}}{k} = k - 1 = \frac{n-2}{2}.$$

□

Definition 5.2.1. *We define that a solution S to Min-ECP clustering is maximal, if S cannot be improved by the merging of any clusters. A strategy for Min-ECP clustering is called maximal, if it always produces maximal solutions.*

Greedy belongs to the class of maximal strategies.

Theorem 5.2.2. *Any maximal online strategy for Min-ECP clustering problem is* $2n - 3$*-competitive.*

Proof. Consider a disagreement edge e connecting vertices v and v' outside any cluster produced by the maximal strategy MAX_n on n vertices. We have two cases: if e is also a disagreement edge in OPT_n, there is a disagreement edge in OPT_n adjacent to v and v'.

Now, if e is not a disagreement edge in OPT_n, then one of v and v' connects to a vertex u, assume it is v, such that the edge $e' = (v, u)$ is a disagreement edge in OPT_n. If no such vertex u exists, then MAX_n would have clustered v and v' in the same cluster, a contradiction, so u does exist.

In this way, we have proved that to each disagreement edge in MAX_n, there must be an adjacent disagreement edge in OPT_n.

Consider now a disagreement edge e in OPT_n. Potentially, all its adjacent edges can be disagreement edges for MAX_n, giving us in the worst case $2n - 4$ adjacent disagreement edges different from e and one where they coincide. Hence the worst case competitive ratio is $2n - 3$. □

From our observation that *greedy* belongs to the class of maximal strategies we have the following corollary.

Corollary *Greedy is* $2n - 3$*-competitive.*

Chapter 6

Conclusions and Future Work

After presenting an overview of the graph theory and online algorithms, we introduced the problem of Online Clique Clustering with objective functions Max-ECP and Min-ECP. The criterion for Max-ECP was to maximise the number of edges inside the cluster and for this purpose two algorithms are described, Greedy and Lazy.

The natural *greedy* approach for Max-ECP clusters vertices into groups by merging every newly arrived vertex with the biggest cluster that it is possible to merge it with. We prove that this approach is at best inversely proportional to the number of vertices in the input ($\frac{1}{(|V|-2)}$).

The Lazy approach on the other hand follows the *greedy* approach, however, at a certain point starts to behave differently - Lazy clusters the vertices initially into singleton clusters and at a moment, when a threshold is meat, merges the newly arrived vertices into clusters that maximise the objective function (bringing the most *profit*). The competitive ratio of Lazy (compared to an optimal offline version OPT) that we achieved and proved is $9/(154 + 16\sqrt{61}) \approx 0.032262$. We simulated both Greedy and Lazy on the example graph on figure 4.6 where *Greedy* scored a profit of 168 whereas Lazy resulted in 188 edges inside clusters. A difference of 20 is significant and should not be ignored.

We have proved a lower bound for Min-ECP and showed that there cannot exist a deterministic strategy that would be better than $\frac{n^{1-\epsilon}}{2}$-competitive, for any $\epsilon > 0$. Additionally, we proved that *greedy* for Min-ECP is no better than $\frac{(n-2)}{2}$-competitive. Moreover, we showed that *greedy* belongs to a class of Maximal

Strategies and any strategy that is Maximal is at most $2n - 3$-competitive.

The next step is to identify possible modification of the strategy and parameters in order to achieve a better competitive ratio where the criterion is to maximise the number of edges inside clusters (Max-ECP).

A Lower Bound for Online Min-ECP Clustering has been proven in Section 5.1 to be linear for every deterministic strategy, however, not much has been done to investigate developing a non-deterministic strategy for both Min-ECP and Max-ECP.

On the applied side one can find challenges in Genetics where Clustering is being used. One such example is CRISPR[10] where Lazy could be used to map the responses of proteins that act on genes into groups. CRISPR stands for clustered regularly interspaced short palindromic repeats, which are oddly repetitive stretches of DNA found in bacteria and archaea. This sequences interact with proteins and form a microbial defence system against viruses. Additionally, one of the most promising applications of gene expression analysis is the classification of tissue types according to their gene expression profiles where Lazy can be used to classify types of cancer by mapping similar types into clusters. In cancer related studies, the data consist of expression levels of many thousands of genes in different types of tissues, both benign and malign. By translating this problem to a graph we can investigate, if Lazy can be used as a classifier.

Bibliography

[1] Ben-Dor, A., Shamir, R., Yakhini, Z. *Clustering Gene Expression Patterns.* Journal of computational biology, 1999.

[2] Alvey, S., Borneman, J., Chrobak, M., Crowley, D., Figueroa, A., Hartin, R., Jiang,G., Scupham, A., T., Valinsky, L., Della V., Yin, B. *Analysis of bacterial community composition by oligonucleotide fingerprinting of rRna genes.* Applied and Environmental Microbiology 68, pages 3243-3250, 2002

[3] Mathieu, C., Sankur, O., Schudy, W. *Online Correlation Clustering.* Proc. 7th International Symposium on Theoretical Aspects of Computer Science (STACS 2010), pp. 573–584, 2010.

[4] Cormen, T. H., Leisserson, Charles E., Rivers, R. L. *Introduction to Algorithms (3rd ed.).* MIT Press and McGraw-Hill, 2009.

[5] Bansal, N., Blum, A., Chawla, S. *Correlation Clustering.* Machine Learning 56(1–3), pages 89-113, 2004.

[6] Dessmark, A., Jansson, J., Lingas, A., Lundell, Eva-Marta, Persson, M. *On the Approximability of Maximum and Minimum Edge Clique Partition Problems.* Australian Computer Society, CATS 2006.

[7] Schaeffer, S. E. *Graph clustering.* Computer Science Review 1, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, 2007.

[8] Albers, S. *Online algorithms: a survey.* Mathematical Programming, Springer-Verlag volume 97, Issue 1-2, pp 3-26, 2003.

[9] Fabijan, A., Nilsson, B. J., Persson, M. *Competitive Online Clique Clustering.* CIAC 2013.

[10] Gorman, C., and D. F. Maron. *How RNA Discoveries Are Radically Changing Gene Therapy and Other Medical Treatments.* Scientific American Apr. 2014, pp 39-45.

[11] Kononenko, I., Kukar, M. *Machine Learning and Data Mining: Introduction to Principles and Algorithms.* Horwood Publishing Limited, 2007.

[12] Sleator, D. D., Tarjan, Robert E. *Amortized efficiency of list update and paging rules.* Communications of the ACM 28, 1985.

[13] Awerbuch, B., Kutten, S., Peleg, D. *Competitive distributed job scheduling.* Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, 1992.

[14] Giotis, I., Guruswami, V., *Correlation Clustering with a Fixed Number of Clusters.* Proceeding SODA '06 Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pages 1167-1176

[15] Shamir, R., Sharan, R., Tsur, D. *Cluster graph modification problems.* In Proc. of 28th Workshop on Graph Theory (WG), pages 379–90, 2002.

[16] Kodek, D. *Arhitektura računalniških sistemov*, Ljubljana, Bi-tim, 1994

[17] Smith, A. J. *Design of CPU cache memories.* ACM Computing Surveys (CSUR), pages 473–530, 1982.

[18] Arpaci-Dusseau, R. H., Arpaci-Dusseau, C. A. *Operating Systems: Three Easy Pieces.* Arpaci-Dusseau Books, 2014

[19] Fielding, R. E., Reschke, J. E. Request for Comments: 7230, available at: `http://tools.ietf.org/html/rfc7230#section-6.3`. June 2014.