Software acceleration of Graph Neural Networks

Graph-Dependent Analysis of a Software Acceleration Technique for Graph Neural Network Computation

Informatics engineering

Specialization in Computing

Author: Eloi Campeny Roig

Director: Sergi Abadal Cavallé (Computer Architecture)

GEP Tutor: Joan Sardà Ferrer

1st of July 2022

Contents

List of Figures

List of Tables

Abstract Català

El camp de recerca de les xarxes neuronals aplicades a grafs (GNN) ha experimentat un creixement explosiu en els darrers anys. Aquesta és una conseqüència de la seva capacitat única per gestionar dades en forma de graf, una estructura de dades molt comuna que s'aplica àmpliament a innombrables indústries, des de la sanitat fins als sistemes de recomanació. Malgrat la popularitat, encara hi ha alguns aspectes que no s'han desenvolupat tant com d'altres. Un dels casos és la recerca de maneres més eficients de computar-los.

Aquest problema pateix d'una complexitat important, a causa de la flexibilitat dels grafs per adaptar-se a una àmplia gamma de problemes. Les dissimilaritats estructurals entre diferents grafs dificulten el disseny d'un algorisme eficient amb una bona generalització. Juntament amb això, també hi ha la qüestió de la quantitat considerable de multiplicacions de matrius esparses que afecten negativament a l'eficiència.

S'han proposat diversos nous algorismes d'acceleració, inclosos els enfocats al software, hardware i el hardware-software. Hi ha acceleradors que es basen a representar el graf d'entrada d'una forma més adequada per al hardware per tal d'aconseguir l'acceleració. Altres utilitzen un esquema diferent per calcular el graf per aprofitar diferents plataformes hardware.

En aquesta tesi suggerim dues hipòtesis. A la llum del fet que no tots els acceleradors funcionen bé per a tots els grafs, la nostra primera hipòtesi és que no hi ha cap accelerador que funcioni bé per a tots els grafs. La segona hipòtesi és que, donada una caracterització adequada del graf, podem predir quin pot ser el millor accelerador entre una selecció d'ells.

Per donar una resposta a ambdues hipòtesis, comparem dues alternatives de càlcul GNN: deep graph library (DGL) com a model de referència i TC-GNN com a accelerador de software per a GNN. La comparació es fa a través d'un conjunt de grafs tan ampli i complet com sigui possible, que contingui grafs de moltes estructures diferents. Amb tal de complir aquest objectiu, utilitzem un generador de graf sintètic, GraphLaxy, del nostre grup de recerca. Usarem tots els grafs sintètics per enregistrar el temps que triga DGL i TC-GNN. Amb aquestes dades, entrenem un model per predir el temps d'execució d'ambdues alternatives en funció de les característiques dels grafs d'entrada. Després de tots els experiments, hem descobert que, en general, TC-GNN té un millor rendiment que DGL, especialment per a gràfics petits, mentre que DGL supera en grafs grans i ultra esparsos.

Abstract Castellano

El campo de investigación de las redes neuronales aplicadas a grafos (GNN) ha experimentado un crecimiento explosivo en los últimos años. Esta es una consecuencia de su capacidad única para gestionar datos en forma de grafo, una estructura de datos muy común que se aplica ampliamente a innumerables industrias, desde la sanidad hasta los sistemas de recomendación. A pesar de la popularidad, todavía existen algunos aspectos que no se han desarrollado tanto como otros. Uno de los casos es la búsqueda de maneras más eficientes de computarlos.

Este problema sufre de una complejidad importante, debido a la flexibilidad de los grafos para adaptarse a una amplia gama de problemas. Las disimilitudes estructurales entre distintos grafos dificultan el diseño de un algoritmo eficiente con una buena generalización. Junto a ello, también está la cuestión de la cantidad considerable de multiplicaciones de matrices poco densas que afectan negativamente a la eficiencia.

Se han propuesto varios nuevos algoritmos de aceleración, incluidos los enfocados al *software*, *hardware* y *hardware-software* . Hay aceleradores que se basan en representar el grafo de entrada de una forma más adecuada para *hardware* para conseguir la aceleración. Otros utilizan un esquema diferente para calcular el grafo para aprovechar distintas plataformas *hardware*.

En esta tesis proponemos dos hipótesis. A la luz de que no todos los aceleradores funcionan bien para todos los grafos, nuestra primera hipótesis es que no existe ningún acelerador que funcione bien para todos los grafos. La segunda hipótesis es que, dada una adecuada caracterización del grafo, podemos predecir cuál puede ser el mejor acelerador entre una selección de ellos.

Para dar una respuesta a ambas hipótesis, comparamos dos alternativas de cálculo GNN: *deep graph library* (DGL) como modelo de referencia y TC-GNN como acelerador de *software* para GNN. La comparación se realiza a través de un conjunto de grafos lo más amplio y completo posible, que contenga grafos de muchas estructuras diferentes. Con el fin de cumplir este objetivo, utilizamos un generador de grafo sintético, GraphLaxy, de nuestro grupo de investigación. Usaremos todos los grafos sintéticos para grabar el tiempo que tarda DGL y TC-GNN. Con estos datos, entrenamos un modelo para predecir el tiempo de ejecución de ambas alternativas en función de las características de los grafos de entrada. Después de todos los experimentos, hemos descubierto que, por lo general, TC-GNN tiene un mejor rendimiento que DGL, especialmente para gráficos pequeños, mientras que DGL supera en grafos grandes y ultra dispersos.

Abstract English

The research field of Graph Neural Networks (GNNs) has been experimenting an explosive growth in the recent years. This is a consequence of their unique capability to manage graph data, a highly common data structure which is widely applied across innumerable industries ranging from healthcare to recommendation systems. Despite the popularity, there are still some aspects which have not been developed as much as others. One of the cases is the research on more efficient ways of computing.

This problem suffers from a significant complexity, due to the flexibility of graph for adapting a wide range of problems. The structural dissimilarities between different graphs makes it difficult to design an efficient computing algorithm with good generalization. Along with that, there is also the problem of the considerable amount of sparse matrix multiplications which impact negatively on efficiency.

Several new acceleration algorithms have been proposed, including software, hardware and hardware-software co-design approaches. There are accelerators which rely on represent the input graph in a more suitable form for hardware in order to get the speedup. Others use a different schema of computing the graph for taking advantage of different hardware platforms.

In this thesis, we propose two hypotheses. In light of the fact that not all accelerators work well for all graphs, our first assumption is that there is no accelerator that works well for all graphs.The second hypothesis follows that, given a proper graph characterization, we can predict which can be the best accelerator among selection of them.

Towards providing an answer to both hypotheses, we compare two GNN computing alternatives: deep graph library (DGL) as a baseline library and TC-GNN as a software accelerator for GNNs. The comparison is done over a wide and comprehensive graph dataset which contains graphs of many different structures. To that end, we use a synthetic graph generator, GraphLaxy, from our research group. We train all the graphs with synthetic data and record the time taken by DGL and TC-GNN. With

this data, we train a model to predict the runtime of both alternatives based on the input graph characteristics. After all experiments, we have discovered that in general TC-GNN has better performance than DGL, especially for small graphs, while DGL outperforms in large and ultra-sparse graphs.

1 Context and Scope

1.1 Context

1.1.1 Introduction

Machine Learning (ML) has experimented a massive growth in the last decade due to its capacity to solve extremely complex problems. In particular, Deep Neural Networks (DNNs) had expanded into every aspect of our lives like virtual assistant, recommend systems or image processing for medical diagnosis are just a few examples [1].

However, it is well known that not all neural network architectures fit to all problems. The specific architecture of each DNN type of layers is focused on producing a good results in certain tasks. For instance, by not making any assumption about the structure of the data, a multi-layer perceptron are able to master a wide range of tasks at the cost of being less efficient in a concrete task than other most specific DNNs. In contrast, techniques such as Convolutional Neural Networks (CNNs) or Recursive Neural Networks (RNNs) are biased toward extracting knowledge from a grill locality and sequences of data. This makes them a better fit for specific tasks such as image recognition or treatment of temporal signals, with the downside of being incapable of efficiently handling data with arbitrary structures [2].

In light of the above, there has been a recent interest in deep learning techniques able to model graph-structured data . This structure is inherent in a huge number of problems in the field of complex systems in general. In essence, Graph Neural Networks (GNNs) adapts their structure in order to handle an input graph and, through an iterative process of aggregation of information across vertices. This allows to predict properties for specific nodes, connections, or the graph as a whole, and generalize to unseen graphs [2].

In recent years have seen a rapid increase in research activity in the field of GNNs [3]. Great efforts have been devoted toward improving the efficiency of algorithms, especially for massive graphs, and toward demonstrating their efficacy in a plethora of application. However, less attention has been placed on the efficient processing of this new type of neural networks. While the issue has already been extensively studied for CNNs or RNNs, the treatment of GNNs remains largely unexplored [4] [5].

This is because GNNs are relatively novel and pose unique computing challenges, including the need to support dense and extremely sparse operations for different graph structures, scale to very large graphs and adapt the computation for handling in an efficient way a huge number of GNN algorithm variant. Even though advances in sparse/irregular tensor processing [6] and graph processing may prove useful in accelerating GNNs, addressing their unique computing challenges requires more specialized efforts. Some attempts have been made from a software standpoint, adapting GNN operations to better match the capabilities of processors or graphic units [4].

The Barcelona Neural Networking Center (BNN-UPC) [7] as an initiative of Prof. Albert Cabellos and Prof. Pere Barlet at UPC (Universitat Politècnica de Catalunya) with the main goals of carrying fundamental research in the field of Graph Neural Network applied to Computer Networks, and educating and training the new generation of students. This end degree project is part of this initiative, emphasizing this issue and possible solutions for carrying in an efficient way the computations of GNN.

1.1.2 Problem to be Resolved

The current problem is that we do not know which GNN accelerator method performs better depending on the input graph(s). It would be better to know which accelerator to use depending on the characteristics of the graph(s) we have (number of vertex, number of edges, sparsity, clustering coefficient, etc.).

As stated above, it is crucial to find which software accelerators are more suitable depending on the input graph(s). The aim of this project is to analyze the impact on the time performance during the training process of GNN.

More specifically we compare DGL and TC-GNN time during training process with the same graph. And using the graph characteristics for training a machine learning predictor in order to choose the best form of computing the input graph.

1.1.3 Stakeholders

The project has many involved parties, which can be Classified into two different groups depending on the interaction and benefits they have with the project itself.

Firstly, the stakeholders that have direct interaction with the project are the tutor and the researcher. Sergi Abadal Cavallé is the tutor of this project.

Moreover, one area of BNN-UPC research is focused on analyzing and improving the performance of GNN using software accelerators. Thus, he will lead and guide the researcher in the correct development of the project. The researcher, Eloi Campeny Roig, is responsible for planning, developing and documenting the project, as well as doing the experiments, analyzing and writing conclusions.

Secondly, the stakeholders that do not interact with the project, but receive direct benefits can be divided into two more groups: companies, that use the software accelerators in order to improve the performance of the machine learning methods (GNN). For example recommender systems like amazon that need to compute an enormous graph with millions of clients and products and cut result in huge savings energetic and economic. And the scientific community, who gets access to the study realized and can use the information and conclusions for further studies in this area including a reduction of the computation time.

1.2 Contextualization

1.2.1 Previous Studies

In the past few years, there has been notably increasing research in improving the performance of different software accelerators for GNN.

PCGCN [8] found that partitioning the adjacency matrix of the graph and making a different treatment depending on the density of the matrix make a huge impact on the performance of GNN. After partitioning the original adjacent matrix with a heuristic, the accelerator decides based on the matrix density if is better to represent the adjacency of nodes with the adjacency matrix or doing the calculations edge by edge.

Other researchers like TC-GNN [9] have focused on other forms of representing the adjacency matrix in a more compact form. In this case the accelerator takes the original matrix (n x n), an split it by rows (for example by 8 rows). In the next step, the algorithm remove the columns that are all zeros (in this 8 x n sub-matrix). Using this technique the accelerator reduce the size of the matrix blocks that is more suitable for TPUs and GPUs.

There has also been research in proposing more methods that prune [10] the less important edges or others that quantize the features of the nodes and edges [11].

1.2.2 Justification

All the research mentioned above only present accelerators and compare the performance of frameworks baselines like PyTorch Geometric (PyG) [12] and Deep Graph Library (DGL) [13]. However, none of them tries to combine more than one software accelerator in order to choose which one are more suitable depending on the input graph. Moreover, the evaluations made only consider a small selection of graphs which not be representative of the entire graph space.

When dealing with GNN and big graph like Reddit dataset, it is very important to select which accelerator must be used. This selection will directly affect the performance in the training part of the GNN. From our point of view, the characterization of the graph(s) is needed to select which of the software accelerator are more suitable. It could be the case that one software accelerator performs better for a huge graph than the other, but has poor performance for a smaller or more clustered graph. With all this information, the software could make an automatic decision of which accelerator to use depending on the features of the graph.

1.3 Scope

1.3.1 Objective and Sub-Objective

The main objective in this project is to analyze the best of two software accelerators for GNN depending on the characteristics of the input graph(s).

To accomplish this objective, the project has been subdivided in several sub-objectives:

Theoretical Sub-Objective

- Learn how GNN software accelerators work.
- Learn different graph characterization.
- Analyze different GNN software accelerators.

Practical sub-objective

- Get an implementation of accelerations.
- Obtain and characterize graph datasets.
- Extract information from different software accelerators using characterized graphs.
- Make a time predictor of the accelerator.
- Analyze the accuracy of predictors.

1.3.2 Requirements

- Chose the features that can be useful to characterize a graph, but don not have a huge negative effect on the running time.
- Obtain a wide and a diverse graph dataset that are a fair representation of the graphs diversity.

- Train a machine learning model that give a accurate results and do not have a huge negative effect on the running time.
- Use a controlled environment in order to avoid differences made by the execution environment.
- Use good programming practices, with a readable style and least complexity possible.

1.4 Methodology and Rigor of Project Planning

In this section we explain the methodology and rigor of the project planning, the methodology of the experiments is explained in Section 3.

1.4.1 Methodology

The methodology that I will use for the project is the Kanban methodology, whose principal objective is to manage in a general way how the tasks are completed.

In this methodology, it uses cards, where each one represents a task to do. The cards will be on a board with 4 different columns:

- **To do:** Composed of all tasks that have been specified, but have not been started yet.
- **In progress:** Composed of all tasks that are being developed, but there are no functional.
- **Testing:** Composed of all tasks that have been developed but did not pass the tests yet.
- **Completed:** Composed of all finished and tested tasks.

To control the work, we will use Jira, a web application for software project management. It allows us to control the tasks that define a specific part of the project.

This methodology stands out for being very easy to use and adapt to any type of project, as well as a very intuitive management of projects. In addition, this methodology offers a lot of flexibility that are needed in research projects since there can be huge changes during the development of the project.

In this research, each card will represent an atomic part of the project, for instance cars will be one of this type: research (read the papers, read documentation of programming language, etc.), programming (create a controlled environment for tests, create a prediction model or program a series of test), documentation (explain the context of the project, write the methodology or write down the final conclusions of the project) or management (make schedulers, do meetings budgets).

Since there is different types of tasks, some ones like reading papers can not be in the testing column since there are nothing to test. However, When is needed to program a model the card will pass through all the columns.

1.4.2 Project Validation

We will use a GitHub repository as a tool for version control due to the easy access (cloud storage) and the failure recovery, since it stores previous versions of the code. One master branch will include the tested code and one development branch will comprise all the code that is in development or testing stage. The code will be publicly available to the director, so he will have the possibility to follow the work and the results at any time.

In the practical part, runtime measurements will be done in a controlled environment to avoid interferences from other applications running in the same device. Moreover, the entire methodology will be repeated multiple times to obtain a statistical measure of the runtime, as well as validated in a second machine to compare the results. For a fair comparison, it will be checked that all accelerators maintain the same learning rate and output at inference (meaning that they only modify the execution time).

A weekly meeting will be scheduled with the director, with the intention of discussing the project status and checking the tasks to be accomplished the following weeks.

2 State of the Art

Optimizing ML algorithms for high performance and efficiency has seen explosive growth over the past few years. This took place soon after the community discovered the tremendous potential of DNN algorithms and all possible applications. The field of GNNs is arriving at a similar inflexion point. At the time of writing, research on GNN methods was already extensive and is still refining algorithms and exploring new applications with high impact potential [4].

GNN provides a unique set of challenges that have rendered existing libraries and hardware platforms inefficient, including:

1. The existence of multiple GNN variants, which may include edge, vertex, and graph wide updates, with a variety of aggregation and combination functions and possibly incorporating pooling and graph/layer sampling operations as well.These functions affect aspects such as the choice of operations to be accelerated, the complexity of the relative calculation of the aggregation and the combination, or the constraints of order between them and between layers.

2. The dependence of computation on the characteristics of the input graph in terms of size, sparsity, clustering, or the length of the associated feature vectors. The challenge is, therefore, to develop accelerators that can dynamically adapt to the graph characteristics.

3. A unique combination of computing characteristics of deep learning and graph processing. More specifically, the combination often implies MLP-like operations over a dense weight matrix. In contrast, aggregation involves, among other operations, fetching groups of vertices that often lead to irregular memory patterns. Therefore, the challenge is to develop architectures that accelerate such distinct phases and their intertwining at runtime.

4. A wide pool of applications with not only different graph characteristics, but also different performance targets. For example, recommendation systems need to

scale to extremely large graphs of up to billions of edges and target high computational throughput. In contrast, applications such as fraud detection rather need to focus on latency. This highlights the need for acceleration techniques that address huge number of possible scenarios.

A direct consequence of the aforementioned aspects is that the bottleneck or the critical operation may vary across GNNs or applications. In light of these challenges, several libraries have been proposed to improve the support for GNNs and efficiently compute its multiple variants both in inference and training. The extensions of popular libraries such as PyTorch or Tensorflow (TF) are clear examples of this.

2.1 Software Frameworks and Accelerators

The challenges of GNN processing rendered both traditional DNN libraries and graph processing frameworks inefficient. The reason is the alternating computing phases of GNNs. DNN libraries would be good at speeding up combination operations within vertices and edges, but perform poorly during aggregation. Graph processing libraries, instead, do a good job at managing irregular memory accesses when traversing the graph. However, these libraries assume trivial operations at the vertices, which is not the case in GNNs. To bridge this gap, recent works have started investigating how to adapt the libraries to (i) provide easy to program interfaces to implement multiple GNN variants, (ii) handle the variety of potentially sparse GNN operations efficiently in widespread GPU hardware, and (iii) scale computations to large-scale graphs and multiple GPUs [4].

In the following part of this subsection is a brief summary of some GNN frameworks (PyG and DGL) and some accelerators (PCGCN and TC-GNN). However there exist an increasing number of papers on differents forms to tackle the computations of GNN. In this tesis we only explore the accelerations that keeps the accuracy of the models. However exists a pruning algorithms [10][14] or quantifying algorithms [11][15]. A big picture of all kinds of accelerators are exposed at this GNN survey [5]

2.1.1 PyTorch Geometric (PyG)

PyG [12] is a widespread library that is built upon PyTorch and that provides support for relational learning. The key aspect is the definition of a message passing interface with definition of message and update functions for neighbourhood aggregation and combination, respectively, and multiple pooling operations. To accelerate GNN processing, PyG handles sparsity via dedicated GPU scatter and gather kernels that operate in all edges and nodes in parallel, instead of using sparse matrix multiplication kernels. Relevantly, Facebook released Pytorch-BigGraph, a library that allows to process arbitrarily large graphs by introducing partitioning and distributed processing and that could complement PyG.

2.1.2 Deep Graph Library (DGL)

DGL [13] is a recent library that works on top of Tensor Flow (TF), PyTorch, or MXNet, and provides plenty of examples and code for multiple GNNs. The library defines three functions: message for edge aggregation and update and reduce and update for aggregation and combination at the nodes. To boost performance, DGL takes a matrix multiplication approach and leverages specialized kernels for GPUs or TPUs. In particular, both sampled dense-dense and sparse matrix multiplications are considered together with node, edge or feature parallelization. As discussed in their work, DGL uses heuristics to choose among the different options as the optimal parallelization scheme depends on multiple factors including the input graph. Thanks to this approach, DGL claims to achieve an order of magnitude faster training than PyG. Recently, researchers at Amazon have released a DistDGL, a system based on DGL for distributed mini-batch training scalable to billion-edge graphs . To achieve it, DistDGL uses min-cut graph partitioning via a lightweight algorithm.

2.1.3 PCGCN

PCGCN [8], the paper by Tian and co-authors present a partition-centric approach to acceleration of GNNs in GPUs, which they implement on top of Tensor Flow. The contribution is motivated by the power-law distribution of the node degrees in a graph, which largely affects partitioning. PCGCN applies a locality-aware partitioning, METIS, that helps obtaining dense sub-matrices. That, however, does not prevent sparse partitions to appear. To combat this, PCGCN profiles the partitions at runtime and applies a dual-mode of operation: dense matrix representation and multiplication kernels when dense, and column-sparse representation and sparse kernels otherwise. In the paper, the authors compare their implementation with vanilla TF, and also DGL and PyG, and report the lowest speedup across libraries. Even in this case, PCGCN always speeds up execution and achieves up to 8.8× in highly clustered graphs.

2.1.4 TC-GNN

TC-GNN [9] introduce the first TCU-based GNN acceleration design on GPUs. The key insight is to let the input sparse graph fit the dense computation of TCU. Instead of exhaustively traversing all sparse matrix tiles and determine whether to process each tile, TC-GNN use a new sparse graph translation (SGT) technique that identify those non-zero tiles and condense non-zero elements from these tiles into a fewer number of "dense" tiles. Therefore, SGT merges the unnecessary data in order to avoid high-cost memory access. TC-GNN exploits the benefits of CUDA core and TCU collaboration. The major design idea is that the CUDA core which is more excel in fine grained thread-level execution would be a good candidate for managing memory-intensive data access. While TCU which is more powerful in handling simple arithmetic operations (e.g., multiplication and addition) can be well-suited for compute-intensive GEMM on dense tiles generated from SGT. At the framework level, TC-GNN is integrated with Pytorch framework.

3 Methodology of the Experiments

Towards proving both initial hypotheses, there are not an ideal GNN accelerator and depending on the graph characteristic we can predict the best compute platform, several experiments have been implemented. These involve measuring the performance of the two models on the selected datasets. All the experiments carried out to compare the different algorithms consist on the training task using the graph convolution. The experiments have been executed in a server of BNN-UPC group. Concretely, this environment is equipped with an AMD Ryzen 9 3950X 16-Core Processor, an NVIDIA GeForce 3090 with 24 GB G6X GPU and 64 GB of RAM memory. To measure the runtime, the time module from python has been used. The time measured is the average of 200 epochs in the training phase.

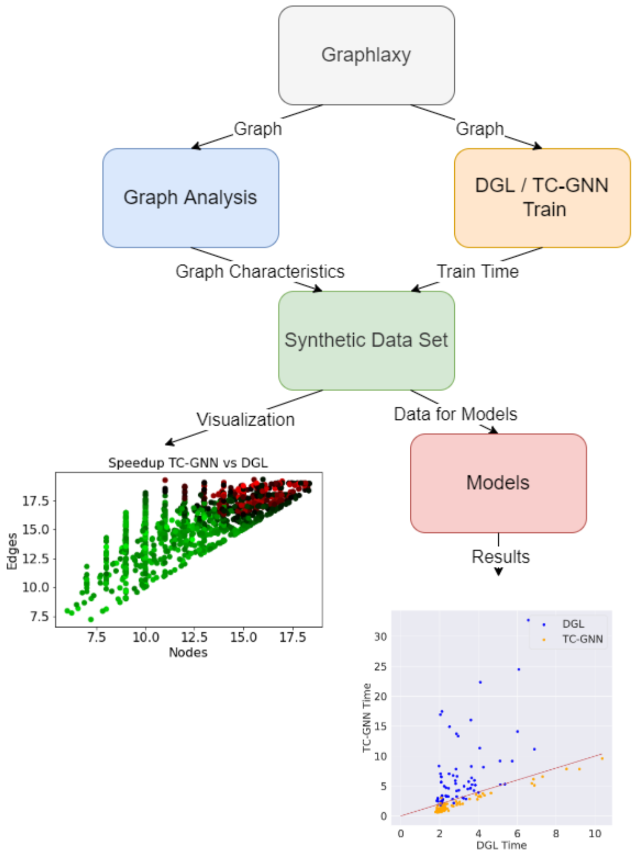


Figure 1: Diagram of the experimental methodology. [Own compilation]

3.1 Datasets

For the experiments, a total number of 14 real datasets and 1 000 synthetic datasets have been used. The real datasets are from TC-GNN original papers and the synthetic are generated using the tool graphlaxy. This dataset has been chosen with the intention of providing the most variety possible to make a solid study of the hypotheses. The results of the characterizations of the graphs (number of nodes, density, average clustering coefficient, etc.) have been obtained via the pertinent functions of NetworkX library. Some examples of the variety of graphs from the real datasets are listed below, and their characteristics of all graphs are shown in Tables 1.

3.1.1 Real Datasets

Here there is some examples of the real graph.

- The **CiteSeer** dataset consists of 3 312 scientific publications classified into one of six classes. The citation network consists of 4 732 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 3 703 unique words [16].
- The **Cora** dataset consists of 2708 scientific publications classified into one of seven classes. The citation network consists of 5429 links. Each publication in the dataset is described by a 0/1-valued word vector indicating the absence/presence of the corresponding word from the dictionary. The dictionary consists of 1433 unique words [16].
- The **Pubmed** dataset consists of 19717 scientific publications from PubMed database pertaining to diabetes classified into one of three classes. The citation network consists of 44338 links. Each publication in the dataset is described by a TF/IDF weighted word vector from a dictionary which consists of 500 unique words [16].

- **Protein-Protein Interaction (PPI)** consists of a set of graph corresponding to a different human tissue. positional gene sets are used, motif gene sets and immunological signatures as features and gene ontology sets as labels (121 in total), collected from the Molecular Signatures Database. The average graph contains 2373 nodes, with an average degree of 28.8 [16].

- **PROTEINS** is a dataset of proteins that are classified as enzymes or non-enzymes. Nodes represent the amino acids and two nodes are connected by an edge if they are less than 6 Angstroms apart [16].

- **Yeast** dataset consists of a protein-protein interaction network. Interaction detection methods have led to the discovery of thousands of interactions between proteins, and discerning relevance within large-scale datasets is important to present-day biology [16].

- **Amazon product co-purchasing network (com-amazon)** was collected by crawling Amazon website. It is based on Customers Who Bought This Item Also Bought feature of the Amazon website. If a product i is frequently co-purchased with product j, the graph contains a directed edge from i to j [17].

In our experiment the dimensions and class are not relevant, we only use the structure of the graph. The calculations are done with random numbers due to that we only need the time and not the accuracy. We only expose these characteristics for giving a general idea of the huge differences between graph. In the Table 1 are some characteristics of all real graphs.

3.1.2 Graphlaxy

Since we need a huge and as representative as possible graph dataset for training the models we use synthetic data. For carrying out this problem of generating graphs we use graphlaxy [18], a new tool that which allows us to generate a massive dataset.

, , , , , , , , Dataset	#Vertex	#Edge	Dim.	#Class	Density	Gini
Citeseer	3 327	9 464	3 703	6	8.45e-04	0.40
Cora	2 708	10 858	1433	7	1.44e-03	0.39
Pubmed	19 717	88 676	500	3	2.28e-04	0.58
PPI	56 944	818 716	50	121	5.05e-04	0.57
PROTEINS full	43 471	162 088	29	2	8.58e-05	0.12
OVCAR 8H	1 890 931	3 946 402	66	2	1.11e-06	0.27
Yeast	1 714 644	3 636 546	74	2	1.24e-06	0.14
DD	334 925	1 686 092	89	2	1.50e-05	0.18
YeastH	3 139 988	6 487 230	75	2	6.59e-07	0.28
amazon0505	410 236	4 878 875	96	22	2.9e-05	0.38
artist	50 515	1 638 396	100	12	6.42e-04	0.65
com-amazon	334 863	1 851 744	96	22	1.65e-05	0.36
soc-BlogCatalog	88 784	2 093 195	128	39	5.311e-04	0.85
amazon0601	403 394	3 387 388	96	22	3.00e-05	0.35

Table 1: Real graph features. [Own compilation]

As it is explained in the original paper, graphlaxy generate a uniform distribution along the clustering coefficient and the logarithm of the density.

The input for graphlaxy are the range between the minimum and maximum number of edges and the number of graphs. In this experiment the edges range between 1 000 and 1 000 000. We chose the minimum of 1 000 edges, because the smaller graph can be computed in no time. The maximum of 1 000 000 are chosen due to the server capacities and time restriction impose. We impose a maximum of 5 minutes to generate a graph, and a good approximation of this is 1 000 000: Without this condition the experiment will take much time. Finally, we chose 1 000 graph because is a huge amount of graph and all tests can be computed in approximate 4 h.

3.2 Features

In order to characterize the graph, we use several metrics that are explained in the next list.

- Number of nodes ($|N|$) : Just the total number of nodes in the graph.
- Number of edges ($|E|$): Just the total number of edges in the graph.
- Number of density: density of a graph is calculated with the next formula $D = \frac{|E|}{|N| \times (|N|-1)}$
- Clustering: is the average of the number trips with size 3 of each node.
- Min Degree: the minimum number of neighbours.
- Q1 neighbours: the number of neighbours of the fist quantile, ordered an increasing number of neighbours.
- Q2 neighbours: the number of neighbours of the second quantile, ordered an increasing number of neighbours.
- Q3 neighbours: the number of neighbours of the third quantile, ordered an increasing number of neighbours.
- Max Degree: the maximum number of neighbours.
- Average Neighbors: the average number of neighbours for each node.

In order to obtain more information from the graph use metrics from other fields of knowledge for a better graph characterization.

- Gini perfect: this metric is generated in order to calculate Gini metric. This metric is approximately #Nodes x #Edges and represents the most equal distribution of the node's degree.

- Gini difference: this metric is generated in order to calculate Gini metric. This metric is at most equal to Gini perfect and represents the real distribution of the graph node's degree.
- Gini: is an economic metrics for inequality, we use this metric to measure the inequality in the node's degree.

3.3 Procedure of the Experiments

First of all we generate 1 000 graph using graphlaxy. With the specified parameters (between 1 000 - 1 000 000 edges). In reality the number of edges is under the specified conditions due to the probabilistic form of generating the graph.

When all 1 000 graphs are generated, we need to extract the metrics of every single graph. As is mentioned in previous sections, this is done with NetworkX framework and specific code programmed by the researcher. The characteristics analyzed can be found in the Section 3.2.

After generating the graph we use the main code for executing all the graph with the same number of: input features (16), hidden features (16), number of classes (2) and number of layers (2). With this experiments, we obtain the times for the TC-GNN and DGL. We save this information in a CSV format for a posterior analysis.

When we have all the tests done on the server, we can proceed with the analysis of the times and the speedup of TC-GNN and DGL.

Before we can move into training phase, first we have to analyze the times of DGL and TC-GNN. In order to take a big picture of the problem, we analyse the speedup depending on the different variables previously extracted from the graph. We plot with scatter-plot the speed up (the color of each point) compared with 2 variables. In this part we are looking for a clearly defined zone, one green (TC-GNN is faster) and a red one (DGL is faster) and a black zone (Booth are even) that divides the other two zones.

For a more specific analysis, we will compare the time of each accelerator with each variable in order to find a patron. If one variable has a clear distribution this means that is an important feature in order to predict the time. If there is not a clear distribution, looks like there are not a correlation between the time and the feature, this means that the characteristic is not the ideal variable (at least not alone).

After all the analysis is done, we can proceed with the training different regressors for predicting the times of TC-GNN and DGL. First, we split the datasets in train (50), validation (20) and test (30). The training data are used for training the model, this will be the only part of the process that the model will be trained. The validation data are used for choosing the best of the different model. And the test data are used for comparing the choices made using both models DGL and TC-GNN.

Secondly, we train different models for predicting the DGL and TC-GNN time. The models that we train and after testes with the validation set are: linear regressor, is the simplest of all models that give us the first view of the difficulty of the problem. A little bit more sophisticated model that we have tested are the K-nearest neighbours, a distance base model, for this reason we rescaled all values in the range of 0-1. In order to find the best parameters we use the function *GridSearchCV*, this function tries different parameters in order to obtain good parameters (probably not the optimal parameters). The first non-lineal model is random forest, the parameters are obtained via the function *RandomizedSearchCV*, this function is similar to *GridSearchCV*, however, this function is faster but give worst result. Ending the list of models are the neural network, the architecture of this net are: four layers (16,8,4,1 respectively output variables) the first three with relu activation function, and the last one with lineal activation function. The optimizer is Adam with an exponential decay learning rate starting in 1e-3 and a decay of 0.8, the optimization variable is the mean square error. These parameters are chosen with the train and error method and the experience of the researcher.

With all of these models trained: linear regressor, K-nearest neighbours, random forest and neural network. We select the best one.

With the both models chosen, we made the final model that for a given characteristics of the graph, calculate the time for DGL and TC-GNN and give the decision of which one is the best for that graph. With this model, we use the test set for obtaining the answers of the final model. With this results we sum of the real times of the selected accelerators and compare with the ideal time (chose always the smallest real time), and the naive decision of choosing only DGL and only TC-GNN.

4 Results and Analysis

4.1 Comparison with the Original Paper

Before we run our experiment, we try to reproduce the original paper experiments. Surprisingly, our results were totally different as the Figure 2 and the Figure 3 show.

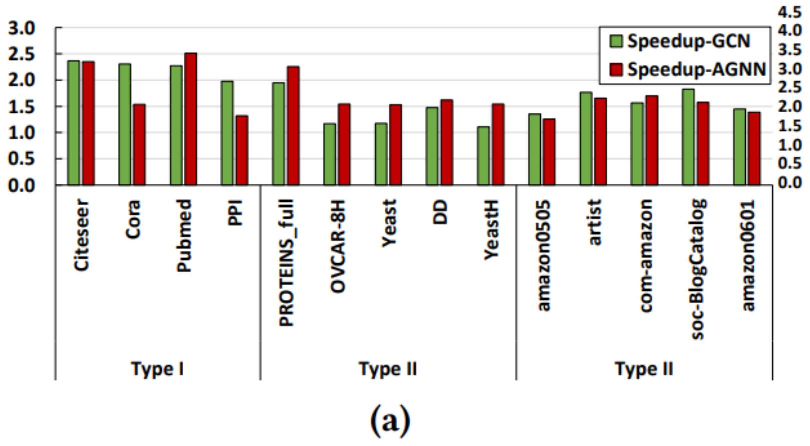


Figure 2: Speedup of TC-GNN [9] respect to DGL [13] extracted from the original paper. [9]

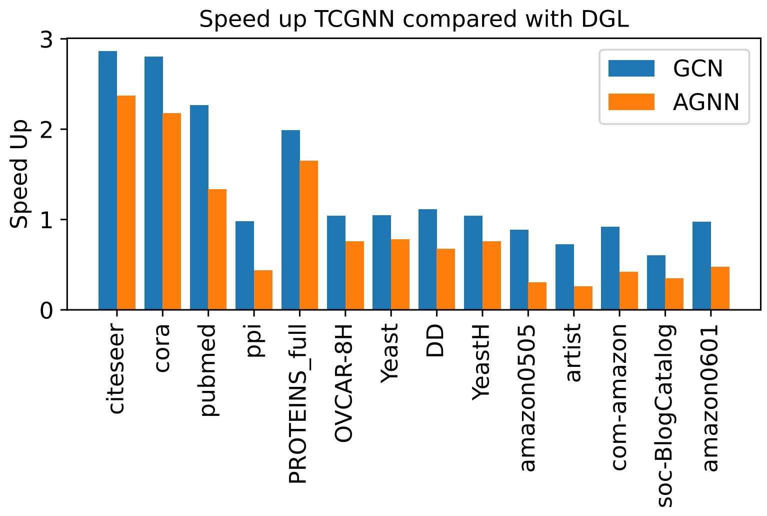


Figure 3: Speedup of TC-GNN [9] respect to DGL [13] obtained from our experiments. [Own compilation]

In the original Figure 2 all the speedups are above one, whereas our results show some cases that have a speedup under one. We think that the difference between our results and the results of the original paper it is produced by some changes in the DGL back-end. But given that we do not have access to the original times (only the speedups) we cannot give a clear answer. The hypothesis of using different hardware can not explain this huge difference. Both experiments are done with the same GPU and both CPUs have approximate the same capacities.

4.2 Visualization

In order to obtain a better perspective of the problem we generate different scatter-plots. The Figure 4 show the speedup of TC-GNN respect to DGL with different features values. We can observe that there are some characteristics that clearly divide the ones with a speed up bigger than one (green) and the once that are below (red).

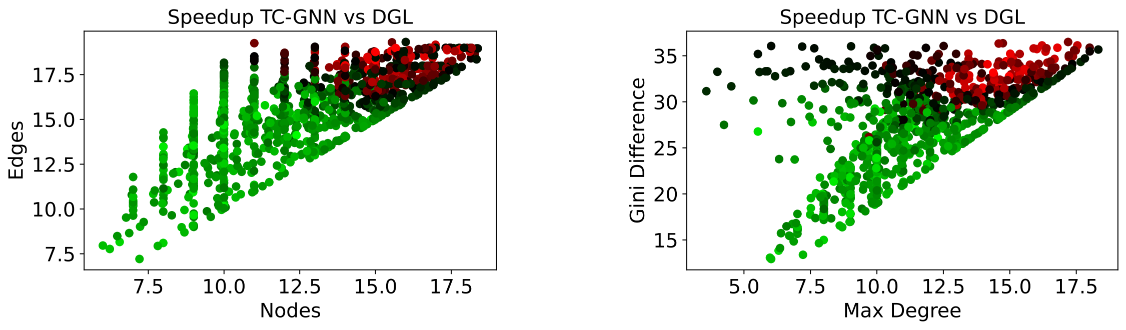


Figure 4: Features vs Speedup in log2 scale. [Own compilation]
Green: TC-GNN is faster than DGL **Red**: DGL is faster than TC-GNN
Black: TC-GNN and DGL are even.

In addition to the previous plots, we generated plots of the time of TC-GNN depending on one feature. The results are plotted in the Figure 5. It is important to notice that the plot of time respect to the edges has a plain part at the star, where there are below 75 000 edges. We suppose that this is produced due to the time that takes to move all the graph information from the CPU to the GPU and the other way, have a bigger impact on the time performance than the effect done by the calculations.

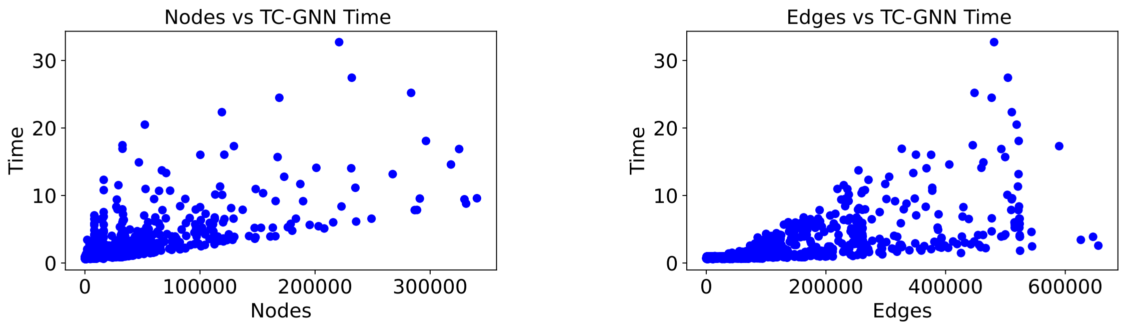


Figure 5: Nodes (left) and Edges (right) vs TC-GNN time. [Own compilation]

With all this plots we can conclude that the are some features that clearly split the space of positive speedup (> 1) and negative speedup (< 1). Moreover, we can start thinking that the flat plane in the TC-GNN time will be a problem for the regressor.

4.3 Models

First, we start with the lineal regressor of DGL, as the images on the left part of Figures 6 show, is a very accurate model (98% using R2 score).

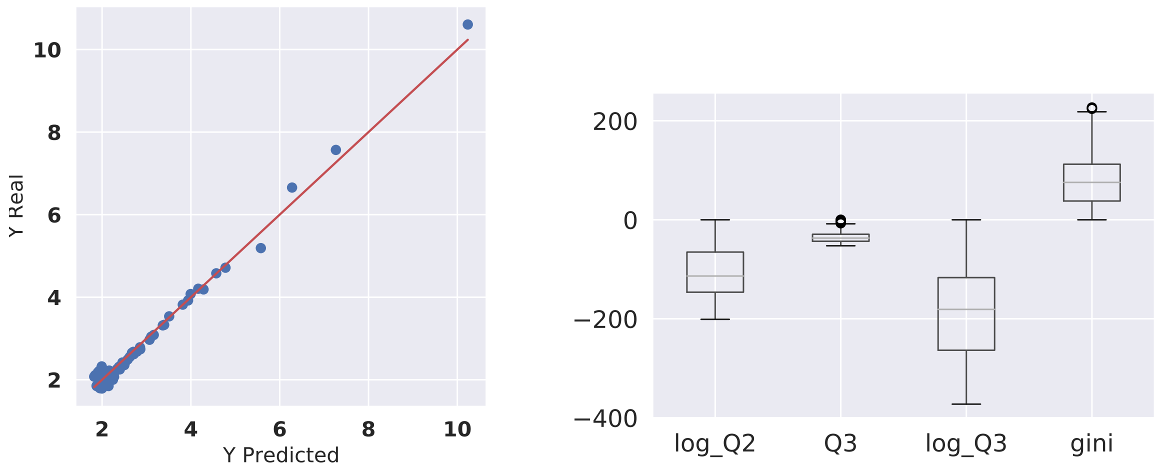


Figure 6: DGL model. [Own compilation]

Left: DGL time prediction using linear regression,

Right: Coefficient importance of more important features from DGL time regression

Moreover, on the right part of Figure 6 reveals the most important features of this model. If the box-plot are near to zero, this features has a small effect on the final decision. In the cases that box-plots are in the positive zone this means that this features makes the GNN computation slower and if there are in negative zones they make the calculus faster.

This feature importance is obtained multiplying the coefficients of the lineal regressor with the trained values. This give has a vector of all the predictions made with the validation set and we represent with this box-plot.

Given the high accuracy of the linear regressor we decide that we will take this model. This decision is made because the linear regressor is the fastest method in inference.

Now we proceed with the TC-GNN model, we use the same method with TC-GNN. However, this time it do not work as well as DGL model, the score is around 52%, the results showed in Figure 7.

When we saw these results we try different methods to obtain a better accuracy. We try KNN, random forest and a neural network as we explained at Section 3.3. The best result was with the neural network with 78% using R2 score, this result are not as good as with the DGL model, but is a huge improve respect the lineal regressor, the results showed in Figure 7.

Like what we did for features of the DGL model, we take the feature importance of the TC-GNN model. This time we use a random permutation in order to infer the importance of each feature. The results are in the Figure 8.

4.4 Final Results

With all models trained, it is time to test the second initial hypothesis, we can predict with is the best form of computing GNN with the graph characteristics. The best approach to see if the hypothesis is true, is measuring the time of the test set with different strategies. First, we calculate the ideal time taking the minimum time between

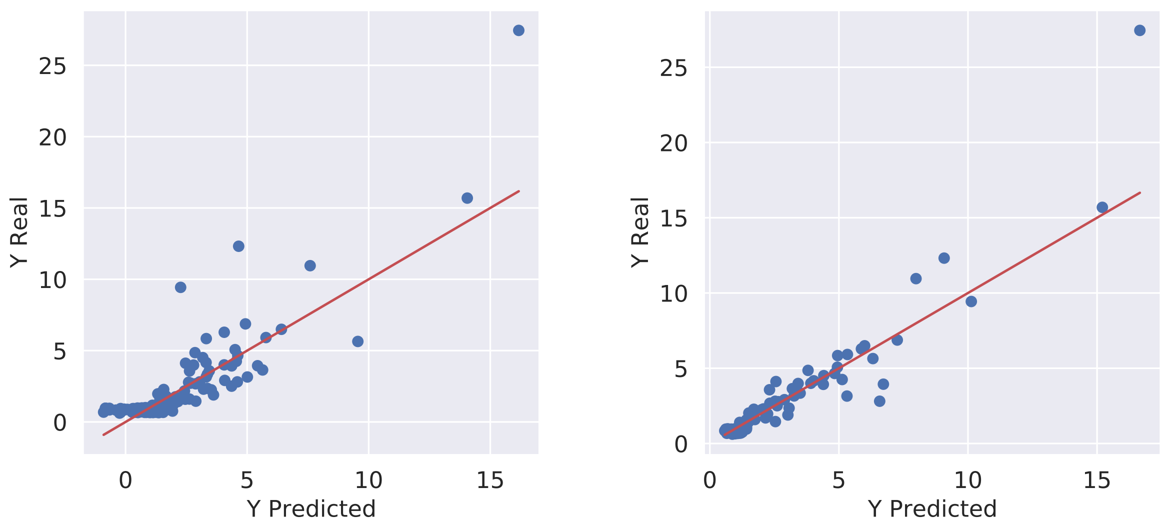


Figure 7: Linear regression and neural network models. [Own compilation]

Left: TC-GNN time with linear regression model,

Right: TC-GNN time with neural network model

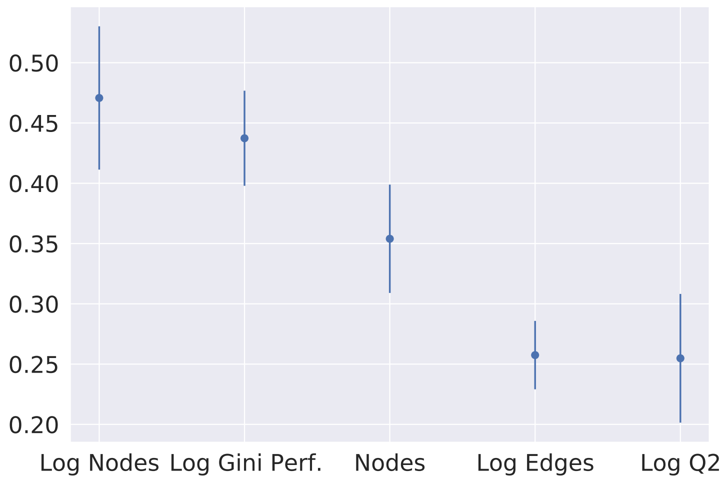


Figure 8: Coefficients of most important features of TC-GNN model. [Own compilation]

DGL and TC-GNN. Secondly, the hypothesis strategies, that is predicting the time with two models and taking the fastest model. And third trying the naive strategies using only DGL and TC-GNN. The model is only used for choosing the model between DGL and TC-GNN, but the times used for comparing the models is always the real time.

The final results of the experiments are Ideal: 499.59s, the Chosen: 507.24s, DGL: 741.87s and TC-GNN: 763.47s, this is shown in the bar plot in the Figure 9. Also there are in the same figure a confusion matrix with the model classifier and real values. As a combination of both graphics, Figure 10 give a general idea of the accuracy of the models.

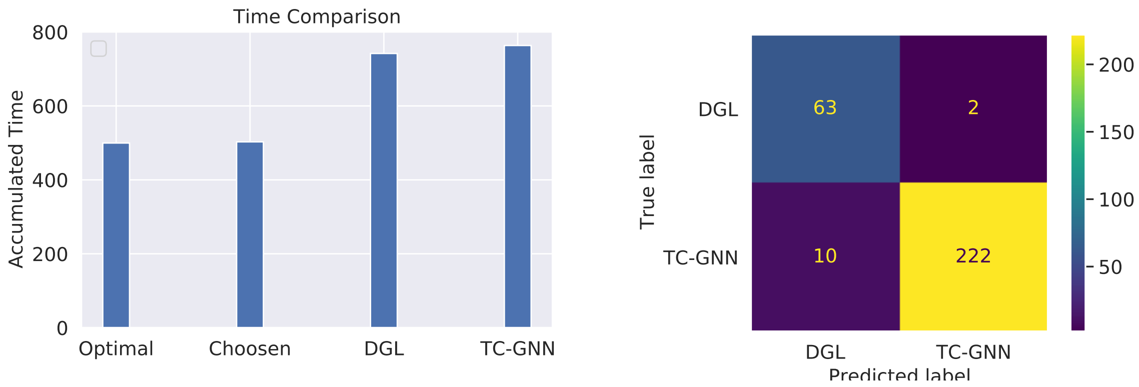


Figure 9: Time and classification of our final model. [Own compilation]
Left: Added time with different strategies, **Right:** Classification of faster accelerator

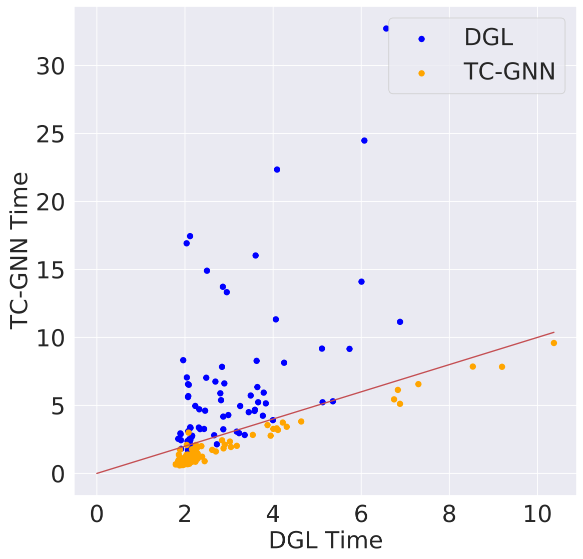


Figure 10: DGL vs TC-GNN time with our model predictions. [Own compilation]

In the bar-plot 9 we can observe that our model is almost the same as ideal time and are much better than the naive models. However the confusion matrix shows that our model have some cases that do not correct classify the graph, but with the last plot unveil that the miss-classified graph needs approximately the same time.

In the Figures 11 and 12 is the same of the above using real graph as a test.

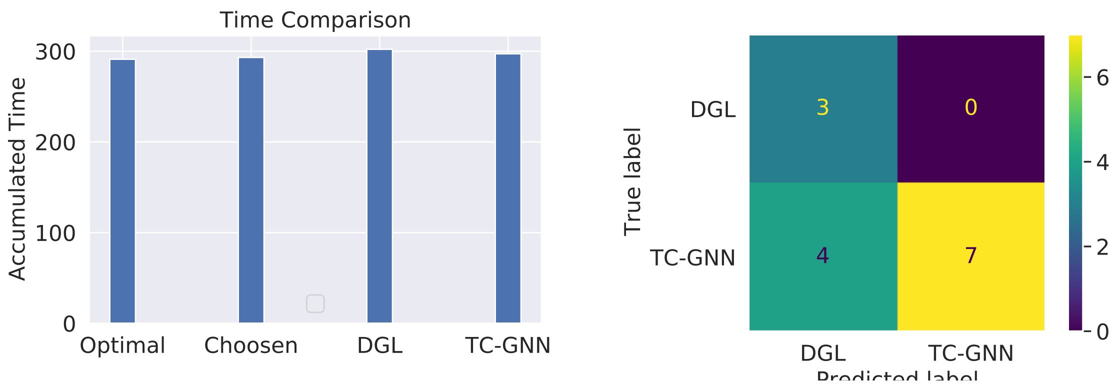


Figure 11: Time and model classification with real datasets. [Own compilation] **Left:** Added time with different strategies, **Right:** Classification of faster accelerator

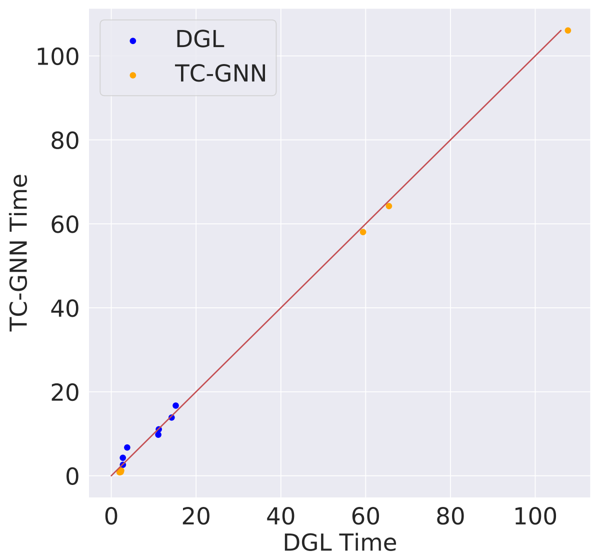


Figure 12: DGL vs TC-GNN time and the classification (real graph). [Own compilation]

5 Conclusions

5.1 Conclusions

From the results of the visualization phase, we can infer that the TC-GNN have a better performance for smaller graph.

As the Figure 10 shows in the synthetic graph our model has very good results, almost the same as the ideal time and produce an approximate 1.5x speedup compared with using always DGL or TC-GNN. Whereas with real datasets the difference between ideal, our model and base lane have approximate the same time. With this result we can not give a clear answer if our model would have a good performance with other real graphs.

5.2 Future Work

In this thesis, we explore among different graph, but we use the same architecture, convolutional neural network with same number layers, input feature size, hidden feature size and number of classes. Moreover the test with real graph do not have enough graphs in order to accept or reject the initial hypotheses.

With all in mind, we propose a study that the same approach will work with different types of GNNs like Attention-based graph Neural Network (AGNN) or Graph Isomorphism Networks (GIN). In addition, with the same architecture with different parameters: number of layers, number of features (input and hidden) or classes. And use more real graphs as a a final test.

In addition, we only use one server, we do not know if the model will generalize for other hardware or all this process have to be done with different hardware.

Finally, we use accelerators that do not lose accuracy, we expect that the same hypothesis can be tackled for the accelerators that lose accuracy. But in this case the experiments should take in mind the accuracy lost in the acceleration process.

6 Project Planning

This project had taken approximately 530 hours, distributed in 140 days starting from February 1st, 2020 until June 22nd, 2020. It was planned to work 3,8 hours approximately every day.

6.1 Task Definition

Following, it is presented all the tasks that will be carried out along the project. For each one, a description is given, together with duration and dependencies with the other tasks. Table 2 summarizes all the information and Figure 13 illustrates the project schedule.

The project management is probably one of the most important group of tasks for the project. It defines the scope of it, the tasks and plans its distribution. Below are shown the multiple tasks for the project management.

- **ICT tools to support project and team management (5h).** We need the latest technology, devices and concepts support the development of a project of this kind. To do so, we have to research different types of software for different types of tasks (e.g. sharing documents and task planning).
- **Context and scope (20h).** We have to indicate the general objective(s) of the project, contextualize it and justify the reason for selecting this subject area.
- **Project planning (10h).** To achieve the project deadline, we need a good planning for all the tasks. This will help us to know in which tasks we have to focus on more and which are the critical ones.
- **Budget and sustainability (15h).** When doing a project, it is very important to know what will be the total cost of it and the impact that will produce its development. Hence, this task focuses on making a budget and analyzing the sustainability of the project.

- **Final project definition (20h).** We have two group the project done in the previous tasks, modifying the parts that were wrong.

- **Meetings (25h).** Face-to-face meetings are scheduled once every week with the tutor of the project. We will discuss the status and the following tasks to carry out. We have added extra time due to possible extraordinary meetings.

But before we can proceed with project management, we have to do some research in order to make an accurate project management.

This project has a big part of research (70h). Hence, before starting the experimental part it is mandatory to do research about previous studies to see the past and recent investigations in multiple software accelerators methods for the GNN (graph neural networks). We will also have to document ourselves in the GNN state of art, as well as the general tools used to program and run GNN like PyG [12] and DLG [13] .

In the theoretical part, we will focus specifically on the paper that propose general accelerators that do not effect the accuracy (or not much) and if possible that have a usable implementation of the algorithm(s). Finally, define for which type of graph can be generalised the methods. This part is divided into three tasks that previously need some research.

- **Check accuracy (10h):** for different types of graph and for every method respect non using accelerators.

- **Compute the expected speed up (30h):** for different types of graph and for every method respect non using accelerators.

- **Generalisation of graph (30h):** for every type of graph and for every method.

Before starting with the experimentation, it is needed to have been done the theoretical part. In addition is needed to adapt the software accelerators to different types of graph. Each algorithm will have to be tested for the correct functioning. And generate a controlled environment. It has been divided also in three tasks.

- **Check accelerators (40h):** in order to detect any bug that produce an undesired result.
- **Adapt methods (15h):** for all types of graph that will be used in the tests.
- **Make a controlled environment (20h):** in order to obtain a fair comparison between different algorithms.

After the experimentation part concludes, we can start with the experimental part.

The experimental and analysis part is the most important one, as this project is oriented to automatically choose the best accelerator depending on the input graph. It has fife different tasks.

- **Obtain the datasets (20h):** to experiment with. It can be divided in two sub-tasks. On the one hand, we can produce a synthetic graph in order to have a huge diversity of the graph. On the other hand, we have to search data sets with all the desired types of graph and have all types of predictions that a GNN can do (node, edge or graph prediction).
- **Experiment (30h):** with the programmed methods and the created and collected data sets and characterize the graph impute. With this information.
- **Generate dataset (20h):** of graph characteristics and running times.
- **Train a predictor (40h):** with the time depending on the graph characterization.
- **Analyze the results (10h):** obtained in the experiments and **draw conclusions**.

Once we have finished will all the previous tasks, we will have to document everything. Firstly, we will have to collect all the information obtained in the experimental and analytical part (15h). Afterwards, we can start writing the documentation of the project (60h).

Finally, we will have to prepare for the oral defense for the presentation of the project. To do so, we will think about possible questions that may come to mind to the senior FIB TFG tribunal members.

6.2 Resources

Every project needs resources to be able to organize it properly and carry out its correct development. These resources have been divided in 4 different groups: human, hardware, software and material resources.

6.2.1 Human Resources

In this project we find three human resources. Firstly, the researcher has the responsibility for the correct development of the project. He will have to plan, experiment, analyze and document the project. On the other hand, the tutor of the project is responsible for leading and guiding the researcher for the correct development of the project. Finally, the GEP tutor is in charge of helping the researcher to do the project management correctly during the first month of the project.

6.2.2 Hardware Resources

One of the essential resources needed is a computer. In this project, it will be Used two different types of personal computers:

- **Personalized desktop computer:** 16 GB of RAM, AMD Ryzen 5 3600 6-Core Processor 3.60 GHz, NVIDIA GeForce GTX 1050 Ti
- **lenovo laptop:** 8 GB of Ram, Intel(R) Core(TM) i5-4200M CPU @2.50 GHz, NVIDIA GeForce 820M

Moreover, we also have to take into account all the resources for the connection to the network (e.g. the router).

6.2.3 Software Resources

We need multiple software resources. Each one will help us in a specific part of the project. To be able to manage the meetings we will use Google Calendar and in case of not being able to meet the tutor in person we will use Google Meet. All the information for the project will be saved in a GitHub repository and on google Colab. The Gantt chart will be created using Ganttproject. We will need a programming language (Python) and some frameworks (PyTorch and PyG) to code the GNN accelerators and the running time predictor. Besides, we will use overleaf as our text editor.

6.2.4 Material Resources

In research projects there is always the need to get knowledge from previous studies and the area in question. In order to obtain this knowledge we will have to read several papers.

6.3 Risk Management

There can be some risks that prevent the correct functioning of the project. Besides, there can appear some obstacles during the execution of the project.

- **Deadline of the project [High risk]:** It can be caused by a bad estimation of the tasks and its duration. As we are doing the planning before having started anything, it is very probable that it will happen. This can be solved by doing a second planning in a more advanced point of the project. We would need to reuse the PC, Ganttproject software, the GEP tutor and the researcher. In case we are running out of time and a new planning does not help, we can still solve this problem by increasing the number of working hours per day.
- **Inexperience in the programming language [Medium risk]:** In this case we decide to use a programming language the researcher has never used, we will have to create a new task of a duration of 25-30 hours that will go before the

programming part. Thus, we will create a new dependency: the tasks on the programming part cannot begin before the end of the ”learning the programming language” task. The new task would have the PC, the programming language and the researcher as resources.

- **The incompleteness of the accelerators [Extreme risk]:** The researcher will probably use a software accelerator that is relatively novel, they cut contain bugs and lack of information. This is a high risk due to the fact that the software accelerators use a plenty of technology's witch the researcher not an expert. For this reason it is so improbable that he can fix a medium/hard bug and I have to chose another accelerator.

- **Back up environment [Medium risk]:** Since for this project is based on time performance, it is important to have an alternative form to execute. If we can not guaranty a controlled environment in Colab, the researcher will have to change the execution environment an execute all the performance test from the beginning.

- **Inexperience in the field [Medium risk]:** Since is the first time the researcher work's in a deep learning and GNN project, it could be the case that the researcher has to spend an 25-30 h in delving on deeper learning and GNN.

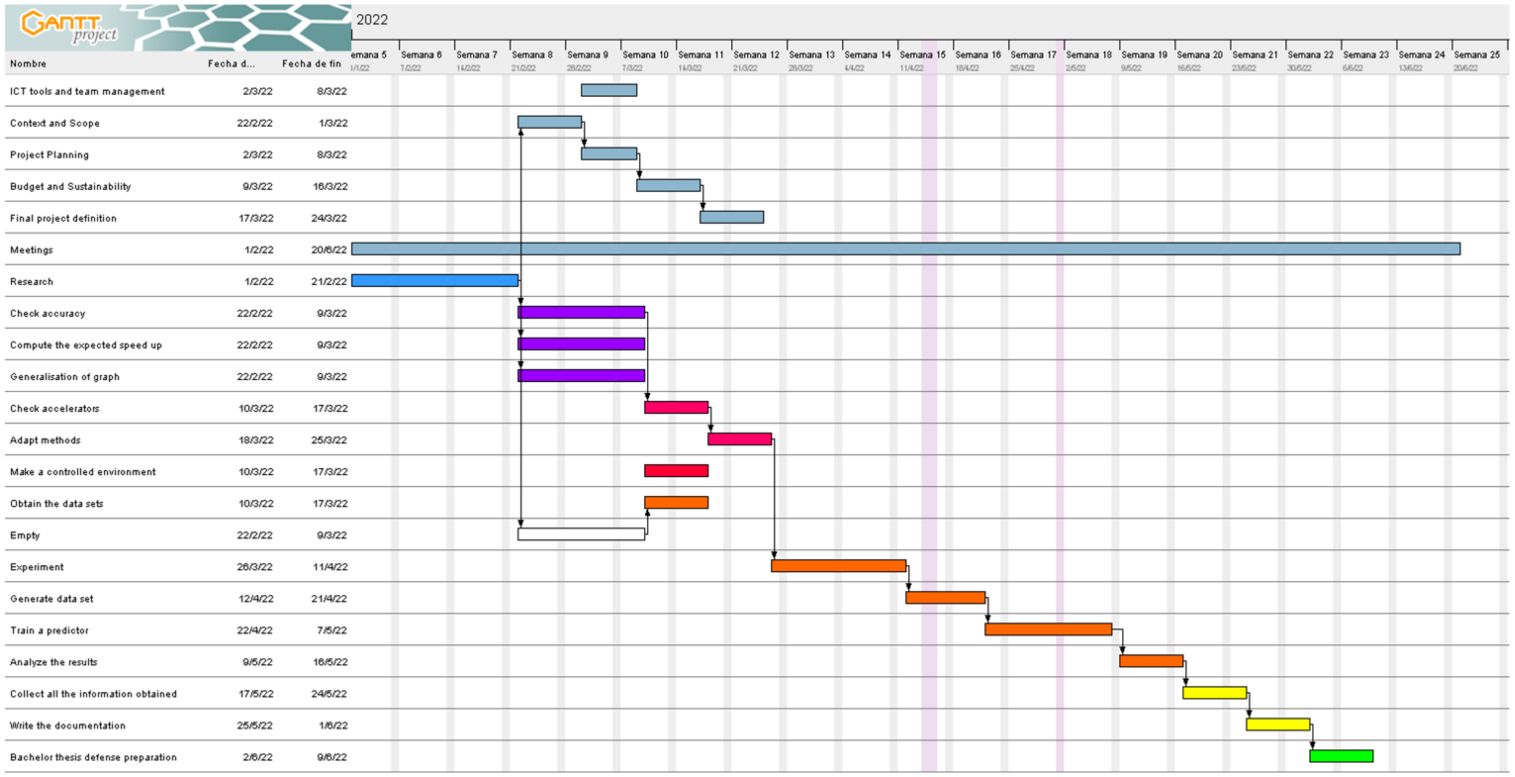
Due to all these risks and obstacles we have overestimated the time for the Meeting task. In case some obstacle appears, we can have an extraordinary meeting to solve it.

ID	Name	Time (h)	Dependencies	Resources
T1	Project management	95		PC, R
T1.1	ICT tools and team management	5		PC, overleaf, R, GEPT
T1.2	Context and Scope	20	T2	PC, overleaf, R, GEPT
T1.3	Project Planning	10	T1.2	PC, overleaf, R, GEPT
T1.4	Budget and Sustainability	15	T1.3	PC, overleaf, R, GEPT
T1.5	Final project definition	20	T1.1,T1.2,T1.3	PC, overleaf, R, GEPT
T1.6	Meetings	25		PC, R, T
T2	Research	70		PC, papers, R
T3	Theoretical part	70		
T3.1	Check accuracy	10	T2	PC, papers, R
T3.2	Compute the expected speed up	30	T2	PC, papers, R
T3.3	Generalisation of graph	30	T2	PC, papers, R
T4	Programming part	75		
T4.1	Check accelerators	40	T3.1	PC, PyG, colab, R
T4.2	Adapt methods	15	T4.1	PC, PyG, colab, R
T4.3	Make a controlled environment	20		PC, PyG, colab, R
T5	Experiments and analysis	120		
T5.1	Obtain the data sets	20	T2	PC, R
T5.2	Experiment	30	T4.2	PC, PyG, colab, R
T5.3	Generate data set	20	T5.2	PC, PyG, colab, R
T5.4	Train a predictor	40	T5.3	PC, Py, colab, R
T5.5	Analyze the results	10	T5.4	PC, Py, colab, R
T6	Project documentation	75		
T6.1	Collect all the information obtained	15	T5.5	PC, overleaf, R
T6.2	Write the documentation	60	T6.1	PC, overleaf, R
T7	Bachelor thesis defense preparation	25	T6.2	PC, R
Total		530		

Table 2: Summary of the information of the tasks. [Own compilation].

GEPT: GEP tutor, T: Tutor and R: Researcher

6.4 Gantt Chart



Nombre	Fecha d...	Fecha de fin
ICT tools and team management	2/3/22	8/3/22
Context and Scope	22/2/22	1/3/22
Project Planning	2/3/22	8/3/22
Budget and Sustainability	9/3/22	16/3/22
Final project definition	17/3/22	24/3/22
Meetings	1/2/22	20/6/22
Research	1/2/22	21/2/22
Check accuracy	22/2/22	9/3/22
Compute the expected speed up	22/2/22	9/3/22
Generalisation of graph	22/2/22	9/3/22
Check accelerators	10/3/22	17/3/22
Adapt methods	18/3/22	25/3/22
Make a controlled environment	10/3/22	17/3/22
Obtain the data sets	10/3/22	17/3/22
Empty	22/2/22	9/3/22
Experiment	26/3/22	11/4/22
Generate data set	12/4/22	21/4/22
Train a predictor	22/4/22	7/5/22
Analyze the results	9/5/22	16/5/22
Collect all the information obtained	17/5/22	24/5/22
Write the documentation	25/5/22	1/6/22
Bachelor thesis defense preparation	2/6/22	9/6/22

Figure 13: Gantt Chart. [Own compilation]

7 Budget and Sustainability

It will be described the elements to consider when doing the budget estimate, which includes personnel costs per task, generic costs and other costs. Moreover, it will be defined management control mechanisms to control the deviations that can appear in the project due to unforeseen obstacles. Finally, it will be answered some questions regarding the sustainability aspect of the project.

7.1 Budget

7.1.1 Personal Costs per Activity

In this section it is computed the total cost for each task defined in previous delivery. The cost for one task will be calculated summing the cost of the work of the personnel. The cost for each worker will be computed multiplying his cost per hour by the amount of time that they will be involved in the activity.

In this project there are 5 types of personnel, each one with a different cost per hour. Firstly, the **project manager** is responsible for the planning and correct development of the project. The **GEP tutor**, the tutor and me will play this role. Secondly, the **programmer** and the **tester** have to program the code and verify its correct functioning. These two roles will be played only by **me**. On the other hand, the **researcher** has to experiment, analyze the results and draw conclusions. **I** will play this role. Finally, there is the **technical writer** who has to document everything that involves the development and results of the project. This role will also be played by **me**. Following, it is shown the annual salary of the different project roles.

Now, we can compute the total cost for each task. Table 4 shows the distribution of time for the personnel for each task, and its total cost. This is known as CPA.

Role	Annual Salary (€)	Total including SS (€)	Price per hour (€)	Role played by
Project manager	45 000	58 500	30.47	GEPT, T, R
Programmer	30 000	39 000	20.31	R
Tester	30 000	39 000	20.31	R
Researcher	35 500	45 500	23.70	R
Technical writer	30 000	39 000	20.31	R

Table 3: Annual salary of the different project roles. [19]

ID	Name	Time (h)	Hours					Cost (€)
			Project manager	Programmer	Tester	Researcher	Technical writer	
T1	Project management	90	90	25	25	25	25	5 010.40
T1.1	ICT tools and team management	5	5	0	0	0	0	152.35
T1.2	Context and Scope	20	20	0	0	0	0	609.40
T1.3	Project Planning	10	10	0	0	0	0	304.70
T1.4	Budget and Sustainability	15	15	0	0	0	0	457.05
T1.5	Final project definition	20	20	0	0	0	0	609.40
T1.6	Meetings	25	25	25	25	25	25	2 877.50
T2	Research	70	0	0	0	70	0	1 659
T3	Theoretical part	70	0	0	0	70	0	1 659
T3.1	Check accuracy	10	0	0	0	10	0	237
T3.2	Compute the expected speed up	30	0	0	0	30	0	711
T3.3	Generalisation of graph	30	0	0	0	30	0	711
T4	Programming part	75	0	35	40	0	0	1 523.25
T4.1	Check accelerators	40	0	0	40	0	0	812.40
T4.2	Adapt methods	15	0	15	0	0	0	304.65
T4.3	Make a controlled environment	20	0	20	0	0	0	406.20
T5	Experiments and analysis	120	0	0	0	120	0	2 844
T5.1	Obtain the data sets	20	0	0	0	20	0	474
T5.2	Experiment	30	0	0	0	30	0	711
T5.3	Generate data set	20	0	0	0	20	0	474
T5.4	Train a predictor	40	0	0	0	40	0	948
T5.5	Analyze the results	10	0	0	0	10	0	237
T6	Project documentation	75	0	0	0	0	75	1 523.25
T6.1	Collect all the information obtained	15	0	0	0	0	15	304.65
T6.2	Write the documentation	60	0	0	0	0	60	1 218.6
T7	Bachelor thesis defense preparation	25	25	0	0	0	0	761.75
Total		530	115	60	65	285	100	14 980.65

Table 4: Summary of the information of the tasks. [Own compilation].

GEPT: GEP tutor, T: Tutor and R: Researcher

7.1.2 Generic Costs

Amortization One aspect to take into account is the amortization of the material resources used in the project. It is considered an average of 3,8 working hours per day during a total of 140 days. It is estimated that an 80% of the project has been carried out using the desktop computer, whereas the other 20% has been done using the laptop computer. The equation to compute the amortization for each resource is the following:

$$\text{Amortization (€)} = \text{Resource price} \times \frac{1}{4 \text{ years}} \times \frac{1 \text{ years}}{273 \text{ days}} \times \frac{1 \text{ days}}{8 \text{ hours}} \times \text{ hours used}$$

The amortizations in this project are only of hardware because all the software used is free to use. The resources will be used a total of 530 hours. The amortization costs are shown below.

Hardware	Price (€)	Time (h)	Amortization (€)
Personalized desktop computer	1 200	424	58.24
Lenovo laptop	800	106	9.7
Total			67.94

Table 5: Amortization costs for the hardware resources. [Own calculations]

Electric cost Regarding the electricity cost, the actual cost of the kWh is 0,2 €[20]. We only count the expenses of the hardware when they are turned on. I should remark that for the desktop computer, we have to add also the monitor expenses. Table 6 shows the individual and total costs of energy consumption.

Internet cost The internet rate costs 50 € per month. Taking into account that the project lasts 5 months and that the working hours per day are 3,8 the internet cost is 5 months $\times \frac{50 \text{ €}}{1 \text{ month}} \times \frac{3.8 \text{ h}}{24 \text{ h}} = 39.58$ €.

Hardware	Power (W)	Time (h)	Consumption (kWh)	Cost (€)
Desktop computer	300	424	127 200	25.44
Lenovo laptop	70	106	7 420	1.48
Total			134 620	26.92

Table 6: Electric cost of the hardware resources. [Own calculations]

Travel cost I have to use the public transport to meet with the tutor once every week. The expected number of travels is twice the number of meetings, therefore I have to do 44 travels during the project. To do so, I have to use the T-Casual of 4 zone [21] , which costs 10 journeys/39.20 €. Consequently, the total cost due to travel is (39.20 €/10 journeys) * 40 journeys = 156.80 €.

Generic cost of the project Table 7 summarizes all the generic costs of the project introduced in the previous sections. The total cost is computed summing the CPA cost with the CG cost (generic cost).

Concept	Cost (€)
Amortization	67.94
Electric cost	26.92
Internet cost	39.58
Travel cost	156.80
CG cost	291.24

Table 7: CG cost in the project. [Own calculations]

7.1.3 Other Costs

Contingencies During the development of the project, it can appear unforeseen events, which take part of our budget. For this reason, it is always necessary to prepare a fund of contingency to be prepared to face these events. For the total cost (CPA +

CG = 15 271.89), we have to add a 15% contingency margin. With this, the computed contingency cost is 2 290.78.

Incidental costs We have also to take into account the cost of applying alternative plans in case unexpected events occur during the course of the project. The alternative plans can be found in Table 8 show the total cost to solve these events. The cost for each incident is computed multiplying the price it would cost by the risk probability that the event occurs.

Incident	Estimated Cost (€)	Risk (%)	Cost (€)
Deadline of the project (20 h)	406.2	30	121.86
Inexperience (15 h)	304.65	20	60.93
Accelerators incompleteness (30 h)	609.30	30	182.79
Back up environmen (10)	203.10	30	60.93
Inexperience in the field (10)	237.00	10	23.70
Total			450.21

Table 8: Incident costs of the project. [Own calculations]

7.1.4 Total Cost

The total cost expected for the project is found in Table 9, computed using all the justified costs calculated in the previous sections.

7.1.5 Management Control

In big projects, it is very probable that the budget and time estimations will not be 100% fulfilled due to obstacles (expected and unforeseen). Consequently, we need to define a model for controlling the potential budget differences.

Every time we finish a task, we have to compute the difference of all the involved costs in it (CPA, CG, contingency and incidents). This difference will be computed

Activity	Cost (€)
CPA cost	14 980.65
CG cost	291.24
Contingency	2 290.78
Incidental cost	450.21
Total	18 012.88

Table 9: Total cost of the project. [Own calculations]

in excel document in order to do an exhaustive analyzes. The following are listed the different indicator formulas for the differences we can have:

- **Human resources difference:** It is caused when the personnel do less or more hours than the expected. We compute this difference as shown below.

$$\text{Human Resources Difference} = \sum_{i \in pit} (Estimated_i - Real_i) \times TotalReal_i$$

  Where pit refers to all personnel involved in that task.

- **Amortization difference:** In case we use the hardware resource less or more time than the expected the amortization cost will vary.

$$\text{Amortization Difference} = \sum_{i \in hr} (Estimated_i - Real_i) \times PriceHour_i$$

  Where hr refers to all hardware resources.

- **Travel cost difference:** Probably the number of meetings will vary from the expected due to unexpected obstacles in the project.

$$\text{Travel Costs Difference} = \sum_{i \in pit} (Estimated_i - Real_i) \times 3.92$$

- **Total cost difference:** Groups all differences in the different tasks. It does not take into account contingencies nor incidents.

$$\text{Total Costs Difference} = \sum_{i \in pit} (Estimated_i - Real_i) \times TotalReal_i$$

Doing this, we can visualize and comprehend easily where and why has been a difference and how much is the difference cost. In case the total cost difference is negative, we will have to use the budget reserved for contingencies.

Finally, we will update a list of incidental costs in case any unforeseen event occurs. In case any of these events happen, we will have to use the budget part reserved for incidents.

7.2 Final Analysis of the Project Management

7.2.1 Risks that have Occurred

In this project he only risks that have occurred is the backup environment. Our initial plan contemplated using google celebratory, but TC-GNN do not work in this environment. As an alternative we use a BNN-UPC server, the consequences derived from this fact is the increase of the cost in hardware that is calculated in the following tables. 11.

A part of this risk, there have not occurred any other, the entire schedule was completed as originally planned.

Hardware	Price (€)	Time (h)	Amortization (€)
Personalized desktop computer	1 200	424	58.24
Lenovo laptop	800	106	9.7
BNN-UPC cervee	2 500	20	5.72
Total			73.66

Table 10: Final amortization costs for the hardware resources. [Own calculations]

Amortization Difference = 5.72 €

Electric Cost Difference = 2.2 €

Final Cost Difference = - 2 733.07 €

Thanks to the initial plan and only one risk that has occurred, we save 2 733.07 €. The main reason this project has ended with this difference in final cost is due to the

Hardware	Power (W)	Time (h)	Consumption (kWh)	Cost (€)
Desktop computer	300	424	127 200	25.44
Lenovo laptop	70	106	7 420	1.48
BNN-UPC cervee	550	20	11 000	2.2
Total			145 620	29.12

Table 11: Final electric cost of the hardware resources. [Own calculations]

Concept	Cost (€)
Amortization	73.66
Electric cost	29.12
Internet cost	39.58
Travel cost	156.80
CG cost	299.16

Table 12: Final CG cost in the project. [Own calculations]

Activity	Cost (€)
CPA cost	14 980.65
CG cost	299.16
Total	15 279.81

Table 13: Final total cost of the project. [Own calculations]

main risk of having problems with the accelerator, do not have occurred.

7.3 Sustainability

7.3.1 Economic Dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project.

The reflection on the estimated cost for the project, as well as the management control, can be found in Section 7.1.1 of the document. It has taken into account the human, hardware, software and material resources. The salary for the personnel has been obtained searching through the internet. Moreover, we have also measured other costs (contingencies and incidental costs) and their effects on the budget.

Regarding Useful life: How are currently solved economic issues related to the problem that you want to address? Nowadays, the economic issue related to the problem is very high, because there is no research that says which accelerator is better depends on the input graph. Consequently, the studies have to program and test the different algorithms and experiment with each one, which costs a lot. Moreover, the focus of the investigations about GNN is on design new types of GNN or problems that are suitable to be solved by GNN.

One way to reduce the economic cost in our project would be reusing some software resource such as the code that have been previously programmed. Doing this, we could reduce the total time for the programming and testing tasks (T4) which would reduce the total economic cost for the project.

How will your solution improve economic issues with respect other existing solutions? As explained in the previous question, in the recent studies and in real life cases, they have to experiment with the different methods to see which one(s) works better for their own graph. This has a very high economic cost. In case we succeed in this project, we should be able to explain and justify which method should be used depending on the multiple cases. This would decrease a lot the economic cost, as they would no longer need to experiment with all the methods and choose the one that gives the best results.

7.3.2 Environmental Dimension

Regarding PPP: Have you estimated the environmental impact of the project? I have estimated the environmental impact of the project will be positive. Since the use of better accelerators reduce the time and energy needed to use by GNN. However, each experiment will be done several times and the average of the results will be the final result. Hence, there will be a high invest of electrical and computational power.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources? As said before, the only resource that could be reused is the programmed code by other researchers. With this, we would not need to develop the program and testing tasks (T4) and we could decrease the total electricity consumption.

Note that we cannot use results obtained in other researches because we have to experiment with our own methodology and hardware. Thus, we cannot reuse resources regarding the experimental part.

Regarding Useful Life: How is currently solved the problem that you want to address? As there is no way to know which method works the best for a particular graph with a specific percentage of missing values, people have to experiment with all the algorithms and then choose the best one. Since the novelty of the GNN huge part of the researchers and companies that use GNN probably do not use accelerators.

How will your solution improve the environment with respect other existing solutions? Our solution will advise which method should be used depending on the characterization of the graph. Thanks to that, the electricity consumption can be decreased, as they should no longer experiment with all the existing algorithms or just choose one which is probably the best option.

7.3.3 Social Dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project? First, this project will help me to introduce in the research world and learn which are the steps needed to do for research of this kind. Secondly, be the principal responsible for a big project will help me to organize myself better and plan the projects correctly. Finally, it will also help me to measure the sustainability aspects of the project.

Regarding Useful Life: How is currently solved the problem that you want to address? Currently, the problem is being solved by just choosing one accelerator with no rigorous information about which one is more suitable for that type of graph or even non using one at all.

How will your solution improve the quality of life with respect other existing solutions? The aim of this project is to reduce the time that people need to compare the performances of the methods by advising which ones should give the best results depending on the characterization. Hence, people would have more time to focus on other important tasks rather than choosing the best method. In addition, for the final users of applications that will use GNN (recommendation system) will have a more faster interaction with the application.

Regarding Useful Life: Is there a real need for the project? There is a real need for the project. Probably, it will help more people that do not have high power computers that can only execute few algorithms in the same computer. People who have high power computers can execute multiple methods at the same time and therefore can evaluate the results very quickly. On the other hand, however, those who do not have high power computers will (like smartphones) be able to select which method(s) is the best for the specific case they have and.

Nevertheless, the results and conclusions will serve to guide people to which methods to choose/experiment. In this way, they will be able to turn away some algorithms and will decrease the power consumption, the economic cost and the total time spent on it.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] P. W. Battaglia, J. B. Hamrick, V. Bapst, *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[4] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing graph neural networks: A survey from algorithms to accelerators," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–38, 2021.

[5] X. Liu, M. Yan, L. Deng, *et al.*, "Survey on graph neural network acceleration: An algorithmic perspective," *arXiv preprint arXiv:2202.04822*, 2022.

[6] S. Dave, R. Baghdadi, T. Nowatzki, S. Avancha, A. Shrivastava, and B. Li, "Hardware acceleration of sparse and irregular tensor computations of ml models: A survey and insights," *Proceedings of the IEEE*, vol. 109, no. 10, pp. 1706–1752, 2021.

[7] "Barcelona neural networking center." (), [Online]. Available: `https://bnn.upc.edu/`. (accessed: 18.03.2022).

[8] C. Tian, L. Ma, Z. Yang, and Y. Dai, "Pcgcn: Partition-centric processing for accelerating graph convolutional network," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE, 2020, pp. 936–945.

[9] Y. Wang, B. Feng, and Y. Ding, "Tc-gnn: Accelerating sparse graph neural network computation via dense tensor core on gpus," *arXiv preprint arXiv:2112.02052*, 2021.

[10] T. Chen, Y. Sui, X. Chen, A. Zhang, and Z. Wang, “A unified lottery ticket hypothesis for graph neural networks,” in *International Conference on Machine Learning*, PMLR, 2021, pp. 1695–1706.

[11] S. A. Tailor, J. Fernandez-Marques, and N. D. Lane, “Degree-quant: Quantization-aware training for graph neural networks,” *arXiv preprint arXiv:2008.05000*, 2020.

[12] “Pytorch geometric.” (), [Online]. Available: `https://pytorch-geometric.readthedocs.io/en/latest/`. (accessed: 18.03.2022).

[13] “Deep graph library.” (), [Online]. Available: `https://docs.dgl.ai/`. (accessed: 18.03.2022).

[14] H. Zhou, A. Srivastava, H. Zeng, R. Kannan, and V. Prasanna, “Accelerating large scale real-time gnn inference using channel pruning,” *arXiv preprint arXiv:2105.04528*, 2021.

[15] M. Bahri, G. Bahl, and S. Zafeiriou, “Binary graph neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9492–9501.

[16] “Papers with code.” (), [Online]. Available: `https://paperswithcode.com`. (accessed: 17.06.2022).

[17] “Snap.” (), [Online]. Available: `https://snap.stanford.edu`. (accessed: 17.06.2022).

[18] A. Wassington and S. Abadal, “Bias reduction via cooperative bargaining in synthetic graph dataset generation,” *arXiv preprint arXiv:2205.13901*, 2022.

[19] “Glass door.” (), [Online]. Available: `https://www.glassdoor.es`. (accessed: 18.03.2022).

[20] “Organización de consumidores y usuarios.” (), [Online]. Available: `https://www.ocu.org/vivienda-y-energia/gas-luz/informe/precio-luz`. (accessed: 18.03.2022).

[21] “Transports metropolitans de barcelona.” (), [Online]. Available: `https://www.tmb.cat/`. (accessed: 18.03.2022).