# TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

Context and Related Works

As the method developed in this work approaches Graph Classification problem using Random Features, we in this chapter introduce Graph networks, how they arose from real world problems, what is the Graph Classification problem, in which applications it can be used, state-of-the-art methods and its limitations, and we finish be stating our contribution.

1.1 Graph Structures and its presence in real world

Graph structures are used to represent a set of objects and the interactions/relations that link between different pairs of these objects, where the nature of these objects and their interactions with each other is application-dependent. Generally in graph structures, objects are reduced to what is called nodes and a relation between two objects is reduced to an edge between the corresponding two nodes.
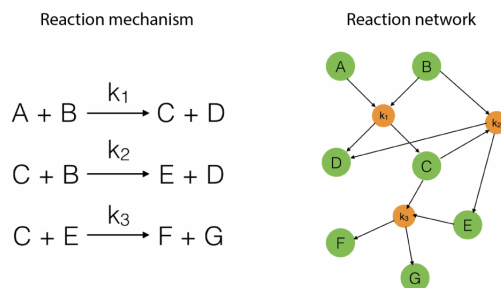
Figure 1.1 Graph structures in representing chemical reactions mechanisms

Network	Nodes	Node Attributes	Edges	Edge Attributes
Transportation System	cities	registered cars	Routes	Length, cost
Banking Network	Account holders	account status	Transactions	Transaction value
Social Network	users	name, country	Interactions	type (like, comment)

Table 1.1 Some real world graphs

The main advantage of using such representation is that unlike using words to describe the relations between objects, Graphs are more aligned with our human brain, it provides us with a better ability to analyse and conclude from the data structures, and actually this is similar to images which are more informative than arrays of pixels values. To better understand how graphs are built in real life, we throw in Table.1.1 some examples of these graphs in different domains.

1.2   Graph classification Problem

There are many varieties of the graph classification problem, where in some cases the task is to do the classification on the Graph level, but in others it is to be done on the nodes level. From another point of view, in some applications each node/edge in the graph dataset has its own features vector that can be used along with the graph structure to classify different graphs into different classes, while in other applications all we have is the graph structure or it happens that some nodes/edges have their features but others don't. In this project, we consider the one case of graph-level classification with the absence of any node/edge features, and that is what we refer to by **_Graph Classification_** from now on unless the opposite is indicated. On the ground, this problem is drastically being addressed in many fields, as in:

- **_Marketing Analytics:_** advertisers and marketers are interested in detecting the influential people communities in Social Networks in the sense that addressing their

products' advertisements to such groups would be a more rewarding investment. This can be approached with graph classification applied on these networks.

- **Banking Security:** graph classification is used to catch unusual patterns of fraudulent transactions.

- **Biology and Genomics:** graphs based on protein-protein interactions are analyzed and classified to determine the stage of evolutionary process of Genes that compound these proteins.

1.3   State-of-the-art Methods for Graph Classification

we here present the traditional algorithms deployed in graph classification problem in its wide sense (with or without node/edge features) which is explained in section 1.2, we simultaneously supply that with the limitations of each of them. In general, these algorithms can be traced back into four main categories: Set based, frequent sub-graph based, kernel based, and Graph Neural Networks based algorithms.

Set based algorithms: this type of algorithms is particularly applied on graphs whose nodes/edges are supplied with features or attributes, so each graph is reduced to a set of nodes, edges or both. Then a distance function of interest between the graphs is computed based on the similarity between pairs of edges/nodes in the corresponding sets. The drawback of this method is that it does not take the structure (Topology) of the graph itself into consideration. For example, if we just compare how much the edges of one graph are similar to the ones of another, we can have two graphs with the same set of edges, which will lead to maximum similarity, but we in reality ignore other important information that can make these graphs completely different such that how many connected communities of nodes these edges form in each graph, how many circles of nodes these edges promote in each graph, etc. On the other hand, a strength point of these algorithms compared to others

is the low computations cost that is usually linear or quadratic in the number of nodes and edges (Shervashidze et al., 2009).

Frequent Sub-graph based algorithms: these algorithms can be applied in two stages, the graph dataset (all the graphs we have in an application) is first analyzed to pick the frequent sub-graphs that occur in different graphs. Then, another analysis is done to choose the most discriminative sub-graphs out of the ones resulted from the first stage. The disadvantage of using this method is the computational cost that grows exponentially with the graph size (Shervashidze et al., 2009).

Graph kernels based algorithms: it is a middle ground between both previous methodologies, where the graph structure (topology) is well considered, and in most cases, these algorithms are designed in a way that the computational time is a polynomial function of the graph size (Shervashidze et al., 2009). However, some effective and competitive kernels still require exponential time, and this is in short the problem we approached in this work using random features to approximate these kernels or to compete them in notably lower computational time.

Graph neural networks (GNNs) based algorithms: GNNs revolutionized learning on graph structures, as it computes a representation vector (embedding vector) for every node in the graph, where this vector is recursively computed by aggregating the representation vectors of neighbor nodes. The goal of this aggregation technique is that nodes that are neighbor (or even close) to each other in the graph are more likely to have a similar representations (with respect to some similarity function) and vise versa. On the graph level, a representation vector is computed by aggregating its nodes representation vectors, and then this vector is used as the features vector which can be fed to a typical Deep Neural Network to learn the classification task. Traditional GNNs such as Graph Convolutional Networks (GCNs) and GraphSAGE failed to provide high performance classifying graphs

(even the ones with simple topology) whose node/edges don't include any original features vectors (Xu et al., 2018). However, another GNN structure was developed to overcome this weakness point, and it is referred to by Graph Isomorphism Network (GIN). Regarding the computational time, it is mainly a matter of the layers number in the network, since this parameter in reality represents how far from a node we want to go in order to compute its representation vector.

1.4 Context and Our Contribution

k-graphlet kernel is one of the aforementioned graph kernels which had proven a competitive performance in graphs classification. Theoretically and empirically, it was shown that a desired performance or a required amount of information to be preserved from the original graph can be reached with sufficiently large k. But, the computational cost is massively large, thus it cannot be applied on large-scale graph datasets. The advent of Optical Processing Units (OPUs) opened a new horizon solving this problem, since it can apply enormous number of *Random Projections* in light speed.

In this work, we did the sufficient mathematical analysis to prove that OPUs' light-speed random feature projections compete the k-graphlet kernel with respect to *Maximum Mean Discrepancy (MMD)* Euclidean metric. Moreover, we empirically tested this hypothesis and made sure that the the theoretical MMD error is aligned with the empirical one with respect to the parameters introduced in the problem (sampling technique, number of sampled subgraphs, number of random features, etc).

REFERENCES

Shervashidze, Nino et al. (2009). "Efficient graphlet kernels for large graph comparison". In: *Artificial Intelligence and Statistics*, pp. 488–495.

Xu, Keyulu et al. (2018). "How powerful are graph neural networks?" In: *arXiv preprint arXiv:1810.00826*.