# State University of Campinas

# Institute of Computation

# 

# 

# Combinatory Study on Graph Network Predictions

# 

# 

# 

# Students:

# Alan Corrales, RA 154475

# Allana Idalgo, RA 145166

# Lucas Racoci, RA 156331

# 

# 

# 

# Professor:

# Prof. Dr. Adín Ramírez

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Campinas,

# June, 2019

# 

# 

# 

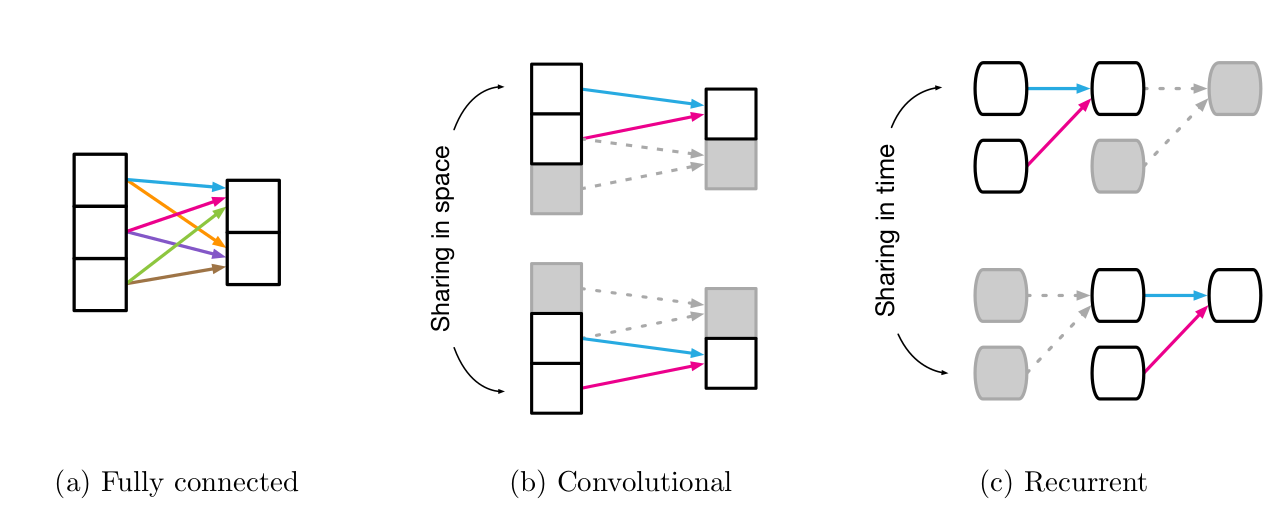
# 

1. **Introduction**
   1. **Abstract**
   2. **Objective**
   3. **Context**
      1. **Artificial Intelligence**
      2. **Neural Networks**
2. **Graph Networks**
   1. **Message Passing Neural Networks**
   2. **Graph Convolutional Neural Networks**
   3. **Graph Independent**
   4. **Graph Nets**
3. **Permutations** 
   1. **Graph Isomorphism Problem**
4. **Methodology**
   1. **Paper Reproduction**
   2. **Problems exploration**
      1. **Shortest Path**
      2. **Hamiltonian Path**
      3. **Physics**
   3. **Adding Permutations**
5. **Results**
6. **Analysis**
7. **Conclusions**

# 

# Introdução

Inteligência Artificial vem passando por uma renascença com grande progresso em visão, linguagem, controle e decisões [1]. Desde 1986 [2] e principalmente a partir de 2006 [3], Redes Neurais Artificiais começaram a se destacar junto com variações como Redes Neurais Convolucionais (CNNs) e Redes Neurais Recorrentes (RNNs).



**Fig. 1: Representação simplificada de alguns modelos de redes neurais [1]**

Recentemente, entre 2017 e 2018, foram publicados os artigos [1],[5],[6] que descrevem algoritmos de predição de grafos generalizados, podendo assim prever, com dada margem de erro, grafos de saída dados grafos de entrada. Isso permitiria aproximar a solução de problemas em variadas áreas, incluindo química molecular, mecânica corpuscular, genética e estudo de comunidades.

Em 2 de Janeiro de 2019, foi feita uma revisão dos principais algoritmos de aprendizagem para a resolução do problema em questão [4]. Essa revisão aborda os três algoritmos citados em [1],[5],[6]. Testes iniciais, motivados pela banca orientadora, indicam a possibilidade de haver uma restrição computacional na generalização desses algoritmos preditivos.

A hipótese é que existem limitações nesses algoritmos por causa da simplificação na representação dos grafos. Grafos são estruturas invariantes a permutações de vértices e todos as técnicas citadas usam conjuntos de treinamentos que não consideram todas as permutações possíveis, porque isso seria impraticável já que o tempo de treino seria fatorial no número de vértices, o que é chamado de explosão combinatória. Para melhor compreender as limitações desses algoritmos mencionados, estes serão avaliados em três datasets diferentes com a finalidade de mapear restrições e possíveis melhorias.

Baseando-se na análise dos experimentos, serão propostos métodos para amenizar o problema que não envolvam o treinamento em todas as permutações dos vértices, o que limitaria os algoritmos em aplicações de entradas pequenas, por conta da explosão combinatória.

Um método que já está sendo hipotetizado para mitigar as limitações é aumentar seletivamente o conjunto de treino para lidar com permutações específicas, que produzam maiores erros na função de avaliação do algoritmo. Apesar da hipótese de melhoria, este trabalho de análise será exploratório, portanto novas estratégias podem surgir a depender dos resultados dos experimentos e do trabalho de pesquisa em torno destes.

# 

# Referências

[[1]](https://arxiv.org/abs/1806.01261) Battaglia et al, “Relational inductive biases, deep learning, and graph networks”, arXiv:1806.01261v3, 2018-10-17

[[2]](http://www.cs.toronto.edu/~fritz/absps/pdp8.pdf) D. E. Rumelhart, G. E. Hinton, R.J. Williams, “Learning Internal Representations by Error Propagation”, p. 317, 1986

[[3]](http://www.cs.toronto.edu/~hinton/absps/fastnc.pdf) G. E. Hinton, S. Osindero, Y. W. Teh, “A fast learning algorithm for deep belief nets”, Neural Computation, vol. 18, 2006

[[4]](https://arxiv.org/abs/1812.08434v2) J. Zhou, et al “Graph Neural Networks: A Review of Methods and Applications”, arXiv:1812.08434v2, 2019-01-02

[[5]](https://arxiv.org/abs/1704.01212) J. Gilmer, et al “Neural message passing for quantum chemistry”, arXiv:1704.01212v2, 2017-06-12

[[6]](https://arxiv.org/abs/1711.07971) X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local neural networks”, arXiv:1711.07971v3, 2018-04-13

# Cronograma

## [Cronograma sugerido](https://docs.google.com/spreadsheets/d/1K8QQMMGayemuGdU6l-k3yffjdm1wEUiL7jPBEiibLCw/edit#gid=0):

1. Estudo inicial, Setup de pipeline e Treinamento (6 semanas)
   1. Leitura bibliográfica
   2. Preparar implementações: Exploração de implementação de protótipos de framework para diferentes algoritmos e modificá-los para aceitarem outros datasets
   3. Exploração da Ferramenta: Salvar e ler dados remotos
   4. Exploração dos datasets: Implementar encoder e decoder para network
   5. Preparar implementações para rodar em um servidor: Configurar um servidor remoto ou um dock com o mínimo para executar os algoritmos da etapa anterior
   6. Adaptar o script para rodar no servidor
   7. Treinar e salvar gradientes: Configurar o treino para salvar checkpoints em nuvem e localmente
2. Teste e Análise (4 semanas)
   1. Estudar permutações a serem realizadas nos dados, se alguma eliminação ou inferência pode ser feita a partir da semântica dos dados: Mostrar e tirar dúvidas
   2. Executar testes: Usando os gradientes de 2.3, testar em uma parte do conjunto de testes para entender os problemas
   3. Análise exploratória dos resultados: Estudar permutações a serem realizadas nos dados, se alguma eliminação ou inferência pode ser feita considerando os resultados anteriores
   4. Executar algoritmos com permutações no conjunto de teste para avaliar o efeito que permutações dos vértices na entrada do conjunto de teste tem na saída da rede treinada em 2.3
   5. Analisar quais permutações são piores para o desempenho da rede: Propor e aplicar métricas e avaliá-las nos resultados dos testes em 3.4
3. Selective Augmentation (4 semanas)
   1. Estudar estratégias de aumentação (augmentation) em grafos que já existam na literatura considerando os resultados de testes já feitos até o momento.
   2. Propor aumentação no conjunto de treino com base nas análise: Com base nos estudos realizados anteriormente, propor e implementar alguma forma de aumentação seletiva, isto é, que leve em consideração todo o conhecimento adquirido até o momento evitando a explosão combinatória
4. Validar proposta (3 semanas)
   1. Retreinar a rede usando as propostas e estratégias implementadas
   2. Analisar os resultados após a estratégia ter sido aplicada e comparar com resultados anteriores para estimar o ganho de eficiência da estratégia implementada

Observação: Cada aluno implementará uma versão para um dataset diferente. O que puder ser utilizado por todos será dividido.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mês** | | | **Fev** | **Março** | | | | | **Abril** | | | | **Maio** | | | | **Junho** | | |
| **Semana** | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| Estudo, Setup e Treino | 1.1 | Leitura Bibliográfica | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.2 | Criar variações das redes | x |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.3 | Salvar e Ler Gradientes |  |  |  |  | x |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.4 | Criar Encoders e Decoders |  | x | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.5 | Preparar servidor | x | x | x | x |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1.6 | Adaptar os scripts |  |  |  |  | x | x |  |  |  |  |  |  |  |  |  |  |  |
| 1.7 | Treinar salvando gradientes |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |  |  |
| Testes e Análise | 2.1 | Estudar permutações |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |  |
| 2.2 | Testar uma parte |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |
| 2.3 | Análise Exploratória |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |  |
| 2.4 | Testar com permutações |  |  |  |  |  |  |  |  | x | x |  |  |  |  |  |  |  |
| 2.5 | Análise de 3.4 |  |  |  |  |  |  |  |  |  | x |  |  |  |  |  |  |  |
| Aumentação | 3.1 | Estudar Aumentação |  |  |  |  |  |  |  |  |  |  | x | x | x |  |  |  |  |
| 3.2 | Propor Aumentação |  |  |  |  |  |  |  |  |  |  |  |  | x | x |  |  |  |
| Validação | 4.1 | Retreinar |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x |  |
| 4.2 | Analisar |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | x | x |

1. **Graph Networks**
   1. **Message Passing Neural Networks**

(Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry.arXiv preprint arXiv:1704.01212.)

There are at least eight notable examples of models from the literature that we can describe using our Message Passing Neural Networks (MPNN) framework. For simplicity we describe MPNNs which operate on undirected graphs G with node features xv and edge features evw. It is trivial to extend the formalism to directed multigraphs. The forward pass has two phases, a message passing phase and a readout phase. The message passing phase runs for T time steps and is defined in terms of message functions Mt and vertex update functions Ut. During the message passing phase, hidden states htv at each node in the graph are updated based on messages mt+1v according to

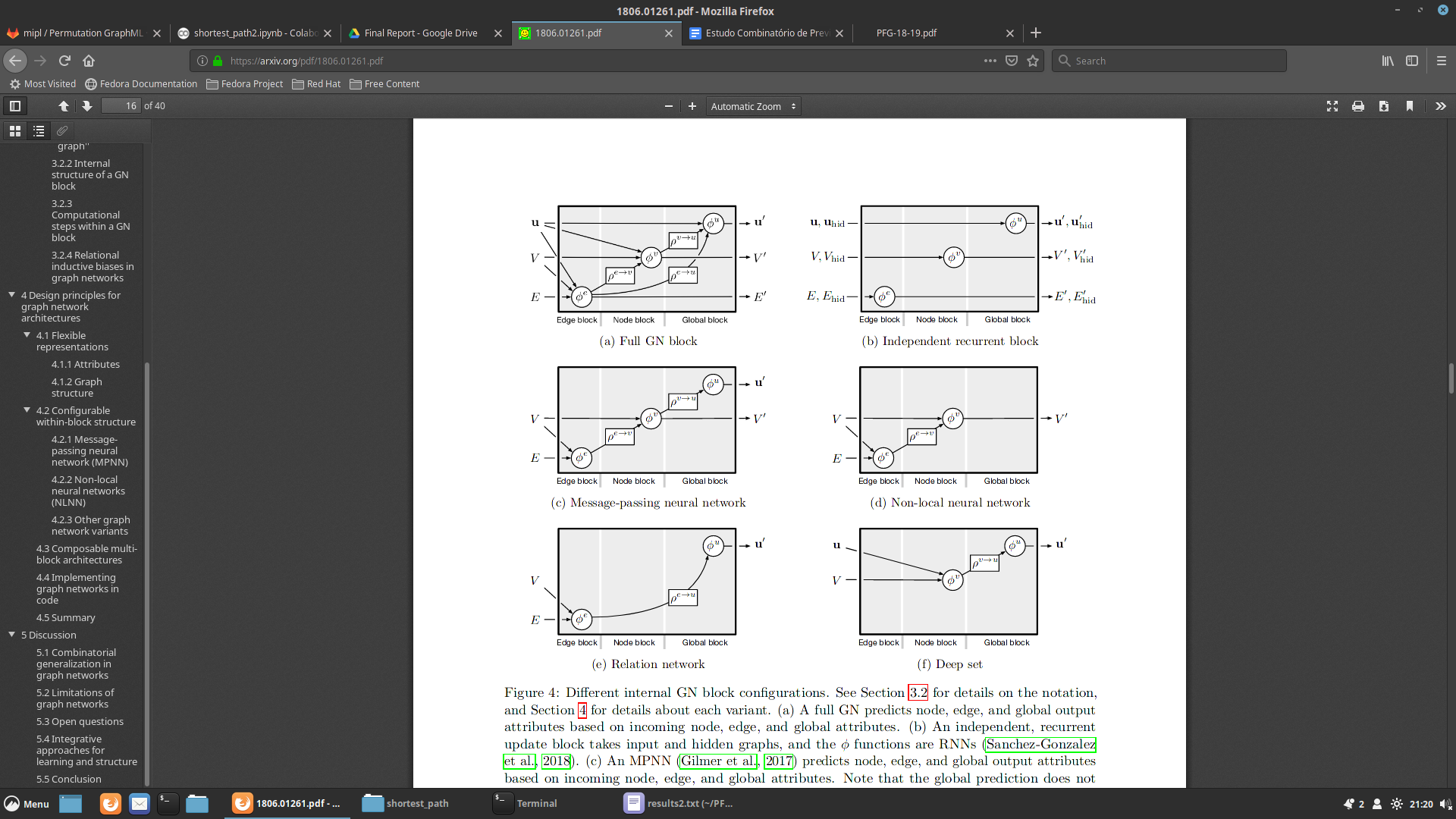
mt+1v=∑w∈N(v)Mt(htv,htw,evw) (1)

ht+1v=Ut(htv,mt+1v) (2)

where in the sum, N(v) denotes the neighbors of v in graph G. The readout phase computes a feature vector for the whole graph using some readout function R according to

y=R({hTv|v∈G}). (3)

The message functions Mt, vertex update functions Ut, and readout function R are all learned differentiable functions. R operates on the set of node states and must be invariant to permutations of the node states in order for the MPNN to be invariant to graph isomorphism. In what follows, we define previous models in the literature by specifying the message function Mt, vertex update function Ut, and readout function R used. Note one could also learn edge features in an MPNN by introducing hidden states for all edges in the graph htevw and updating them analogously to equations 1 and 2. Of the existing MPNNs, only Kearnes et al. (2016) has used this idea.

****

Gilmer et al. (2017)’s MPNN generalizes a number of previous architectures and can be translated naturally into the GN formalism. Following the MPNN paper’s terminology (see Gilmer et al.(2017), pages 2-4):

◦the message function, Mt, plays the role of the GN’s φe, but does not take u as input,

◦element wise summation is used for the GN’s ρe→v,

◦the update function, Ut, plays the role of the GN’s φv

◦the readout function, R, plays the role of the GN’s φu, but does not take u or E′ as input, and thus an analog to the GN’s ρe→u is not required;

◦dmaster serves a roughly similar purpose to the GN’s u, but is defined as an extra node connected to all others, and thus does not influence the edge and global updates directly. It can then be represented in the GN’s V.

* 1. **Graph Convolutional Neural Networks**
  2. **Graph Independent**
  3. **Graph Nets**