

# Graph reduction methods

Yannick Kuhar

May 2024

## 1 Introduction

Data-driven insights can help with the decision-making process in many different domains. Critical domains such as crisis management, predictive maintenance, mobility, public safety, and cyber-security can benefit from this [6]. For these systems to be useful, their predictions must be precise and reliable.

The decision-making models can often be presented as graphs. Due to the complex nature of the relevant tasks, the graphs stemming from them have become large enough to pose storage issues. In such cases, it is prudent to use graph reduction or graph compression methods to decrease the input graph size while keeping all relevant graph properties. Graph reduction further splits into different subfields. Graph coarsening aims to reduce the vertex set and Graph sparsification of the edge set.

This report is split into four sections. Section 2 describes all four graph reduction methods. Section 3 presents our results. Finally, Section 4 gives our concluding remarks.

## 2 Preliminaries

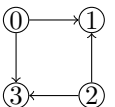
This section explains the concept of graph symmetries and their use in graph compression.

### 2.1 Graph symmetries and symmetry compression

A graph is defined as  $G = (V, E)$ , where  $V(G)$  is the set of vertices and  $E(G)$  is the set of edges. We assume simple graphs with no isolated vertices. Graph vertex-symmetries are a synonym for graph automorphisms. Their standard representation is a permutation of the vertex set, i.e., a bijective function  $\pi : V(G) \rightarrow V(G)$  that preserves connectivity ( $(u, v) \in E(G) \Rightarrow (\pi(u), \pi(v)) \in E(G)$ ). All such permutations form a permutation group  $Aut(G)$  [1], an example is shown in Table 1.

Each permutation of the vertex set induces a corresponding permutation of the edge set, denoted as  $\bar{\pi}$ . Let's take the graph in Table 1 for our example. Vertex symmetry  $\pi = (13)$  induces edge symmetry  $\bar{\pi} = ((0, 1)(0, 3))((2, 1)(2, 3))$ . With

Table 1: On the left is a simple directed graph  $G$  with four vertices and four arcs. All permutations of  $Aut(G)$  are written in standard cycle notation on the right.

G	Aut(G)
	$()$ $(13)$ $(02)$ $(02)(13)$

knowledge of the symmetry  $\pi$ , certain edges become redundant. Specifically, retaining only one edge is sufficient within each cycle  $c \in \bar{\pi}$ . By selecting the first edge from each cycle in  $\pi$ , we obtain the residual graph  $G^\pi$  as depicted in Figure 1.

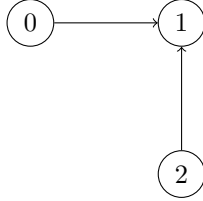


Figure 1: The residual graph after removing all redundant edges.

The pair  $(\pi, G^\pi)$  is an alternative representation of the graph  $G$ , allowing us to derive the entirety of its edge set from this representation.

### 2.1.1 Symmetry compressible graphs

Our interest lies solely in graphs for which the alternative representation is smaller than the original. To this end, we introduce the concept of symmetry compressible graphs, defined as follows:

**Definition 1** (Symmetry compressible graph). *A graph  $G$  belongs to the class SC (Symmetry Compressible), if there exists a permutation  $\pi \in Aut(G)$  such that  $|\pi| + |G^\pi| < |G|$ .*

### 2.1.2 Near symmetry compressible graphs

We can transform non-symmetrical graphs into symmetrical ones by introducing a few modifications. In this context, we restrict these modifications to adding and removing edges. We denote the transformations from graph  $G$  to a symmetry compressible graph  $H$  as  $G \oplus H$ . This leads us to define a new class of graphs:

**Definition 2** (Near symmetry compressible graph). *A graph is considered near*

symmetry compressible, denoted as  $G \in \mathcal{NSC}$  (Near Symmetry Compressible), if there exists a graph  $H$  and a permutation  $\pi \in \text{Aut}(H)$  such that:

$$|\pi| + |H^\pi| + |G \oplus H| < |G|$$

An illustration of a graph belonging to this class is provided in Figure ??.

### 2.1.3 Subgraph symmetry compressible graphs

Now, let's exclusively focus on the removal of edges. Specifically, we narrow our attention to near-symmetry compressible graphs denoted as  $G$ , which encompass symmetry compressible subgraphs denoted as  $H \subseteq G$ . We define this as follows:

**Definition 3** (Subgraph symmetry compressible graph). *A near-symmetry compressible graph is subgraph symmetry compressible, i.e.  $G \in \text{SubSC}$ , if there exists  $H \in G$  such that  $H \in \mathcal{SC}$ .*

### 2.1.4 Graphlet based compression

This algorithm is tailored to operate on graphs from the  $\text{SubSC}$  class. The core principle of compression involves creating a list of graphlets that exhibit symmetry compressibility. For each graphlet  $G'$ , the algorithm identifies the maximal set of subgraphs  $H \subseteq G$  that are isomorphic to  $G'$ .

Subsequently, all such subgraphs are removed from  $G$  and represented as  $(\pi, H^\pi)$ . Once all such pairs have been extracted from the graph, the remaining non-symmetry compressible portion is retained as an edge list. A more detailed description of this procedure can be found in Algorithm 1.

In our experiments, we were constrained by technological limitations, limiting our usage of graphlets with 4, 5, and 6 vertices. Moreover, we exclusively utilized graphlets belonging to  $\mathcal{SC}$ , which amounts to 93921 such graphlets.

---

#### Algorithm 1 Graphlet compression [1]

---

```

1: result = []
2: for  $(\pi, G')$  in Graphlets do
3:   while  $\exists H \subseteq G: H \simeq G'$  do
4:     Let  $\pi_H$  an automorphism of graph  $H$  yielded by  $\pi$ 
5:     result = result +  $(\pi_H, H^{\pi_H})$ 
6:      $G = G \setminus H$ 
7: return result + G

```

---

## 2.2 Graph reduction with spectral and cut guarantees

This method aims to simplify the input graph while keeping its key properties. Let matrix  $L \in \mathbb{R}^{N \times N}$  be positive semidefinite (PSD) representing the structure of graph  $G = (V, E, W)$ . Sets  $V, E$ , and  $W$  represent graphs' vertices, edges,

and weights. Let  $N$  be  $N = |V|$  and  $M = |E|$ . Let  $x$  be some arbitrary vector of size  $N$ .

Loukas et al. define a general graph reduction scheme with the following two equations where  $L_0 = L$  and  $x_0 = x$ :

$$L_l = P_l^\mp L_{l-1} P_l^+$$

$$x_l = P_l x_{l-1}.$$

Where  $P_l \in \mathbf{R}^{N_l} \times \mathbf{R}^{N_{l-1}}$  are matrices with more columns than rows.  $l = 1, 2, \dots, c$  is the level of reduction. Symbol  $\mp$  denotes the transposed pseudoinverse of a matrix.  $N_l$  is the dimensionality at level  $l$  such that  $N_0 = N$  and  $N_c = n \ll N$  [3]. Using this scheme keeps the PSD property, i.e., if input matrix  $L$  is PSD, then so is matrix  $L_c$  reduced using the scheme above. Other properties are kept as well [3].

The paper focused on graph coarsening as a type of graph reduction. Coarsening performs reduction by abiding by a set of constraints [3]. In other words, at each level, this reduction scheme partitions the graph's original vertex set  $v_{l-1}$  and maps it into a smaller vertex set  $V_l$  via a subjective (many-to-one) function. A larger set of vertices mapped to one vertex is called a *contraction set*. Figure 1 shows a simple example of this.

Loukas et al. restricted this process by making it consistent with a graph Laplacian defined as:

$$L(i, j) = \begin{cases} d_{ij} & \text{if } i = j, \\ -w_{ij} & \text{if } e_{ij} \in E \\ 0 & \text{otherwise,} \end{cases}$$

where  $w_{ij}$  is the weight of  $e_{ij}$  and  $d_i$  is the weighted degree (sum of all edge weights) of vertex  $v_i$ . This allows us to keep other matrix properties as we reduce it [3].

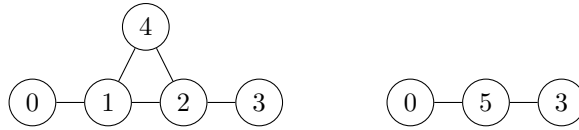


Figure 2: Example of graph coarsening. On the left is the original graph with one contraction set  $c = \{1, 2, 4\}$  that is mapped into vertex 5, resulting in the graph on the right.

### 2.3 Principle of Relevant Information for Graph Sparsification

Graph sparsification reduces the number of edges in a graph while keeping its structural properties. Traditionally, this was done using the Information Bottleneck (IB) formula:

$$\mathcal{L}_{IB} = \min I(X; T) - \beta I(Y; T),$$

where  $I(\cdot; \cdot)$  is the mutual information between two random variables.  $T$  is the object we want to learn from graph node representation  $\{X, Y\}$ .  $\beta$  is the parameter that controls the trade-off between the performance of  $T$  (quantified by  $I(Y; T)$ ) and the complexity of the representation (quantified by  $I(X; T)$ ) [4].

Instead of using IB, Yu et al. explored the use of another information theory concept to achieve graph sparsification, namely the Principle of Relevant information (PRI). Its main benefit over IB is that it requires only one random variable. IB requires the variable  $Y$  and possibly the joint distribution  $P(X, Y)$ . PRI is unsupervised and decomposes  $X$  to obtain  $T$ :

$$\mathcal{L}_{PRI} = \min H(T) + \beta D(P(X) || P(Y)),$$

$H(T)$  is the entropy of  $T$ . This formula reduces the uncertainty of  $T$  and thus finds its statistical regularity.  $D(P(X) || P(Y))$  is the divergence between the distributions of  $X$  and  $T$ .

Formally, graph sparsification takes an input graph  $G = (V, E)$  and constructs a smaller graph  $G_s = (V, E_s)$  such that  $|E_s| \ll |E|$ . Reducing the vertex set is also possible; it is called graph coarsening and is out of scope for this section. This method works with the graph Laplacian  $\rho = D - A$ , where  $D$  is the diagonal degree matrix and  $A$  is the adjacency matrix.

Let  $G$  be the input graph with Laplacian  $\rho$  and  $G_s$  the resulting graph with Laplacian  $\sigma$ . Given the definition of PRI we can formulate our problem as a tradeoff between the entropy  $S(\sigma)$  of  $G_s$  and the dissimilarity  $D(\rho || \sigma)$ :

$$\mathcal{J}_{PRI} = \arg \min_{\sigma} S(\sigma) + \beta D(\rho || \sigma) \text{ [4]}.$$

To solve this equation, Yu et al. defined a gradient descent algorithm. It proved successful as it outperformed the existing approaches and graph neural networks. In its implementation, the model uses two parameters,  $\beta$ , as shown in the equations above, which define the tradeoff between the entropy and the dissimilarity. Parameter  $\alpha$  defines the regularization level during the learning process; if not stated otherwise, it is set to  $\alpha = 0.05$ .

## 2.4 A Graph Coarsening Algorithm for Compressing Representations of Single-Cell Data with Clinical or Experimental Attributes

Graph-based algorithms have become essential in the field of bioinformatics [5]. Many algorithms from this field operate on a graph representation of cell data. In practice, applying these algorithms to extremely large graphs is difficult. To make this possible, we must reduce the graphs to a manageable size while keeping all relevant properties. In other words, we must remove as much redundant information from the graph as possible. One way to do this is to merge nodes into coarse nodes. The procedure is outlined in Algorithm 2

---

**Algorithm 2** Cytocoarsening [5]

---

```

1: Input: feature matrix  $\mathbf{X}$ , attribute vector  $\mathbf{y}$ , number of passes  $\mathbf{p}$ , number
   of KNN neighbours  $\mathbf{K}$ , cutoff parameter  $\alpha$ 
2: Output: coarsened graph  $\mathbf{G}'$ 
3: for 1:P do
4:    $\mathbf{G} = \text{KNN}(\mathbf{X}, \mathbf{K})$ 
5:    $\mathbf{C} = \text{Get.K.Neighbourhoods}(\mathbf{G})$ 
6:    $\mathbf{I}^{\mathbf{C}} = \text{Get.Index.Sets}(\mathbf{C})$ 
7:    $\mathbf{T} = |\mathbf{C}|/4$ 
8:   for  $\mathbf{C}_j \in \mathbf{C}$  do
9:      $c_j^d = \max_{j,k \in I_i^{\mathbf{C}}} \{\|X_{j,:} - X_{k,:}\|_2\}$ 
10:     $c_j^q = \mathbf{y}_{\mathbf{C}_i}^T \mathbf{L} \mathbf{y}_{\mathbf{C}_i'}$ 
11:     $\{\mathbf{T}^q, \mathbf{T}^d\} = \text{Set.Thresholds}(c^q, c^d, \alpha)$ 
12:     $\mathbf{C}_L = \text{Nodes.To.Coarsen}(\mathbf{C}, c^q, c^d)$ 
13:     $\{\mathbf{S}, \mathbf{I}^{\mathbf{S}}\} = \text{Form.Super.Nodes}(\mathbf{C}_L, \mathbf{V}(\mathbf{G}))$ 
14:    for  $i = 1, \dots, |\mathbf{C}_L|$  do
15:       $\bar{\mathbf{S}}_i = \text{Find.Representative}(\mathbf{C}_i^L)$ 
16:       $\mathbf{G}' = \text{Make.Graph}(\mathbf{S})$ 
17:       $\{\mathbf{X}, \mathbf{y}\} = \text{Update.Xs}(\mathbf{X}, \mathbf{y}, \mathbf{I}^{\mathbf{S}})$ 

```

---

The authors use the KNN algorithm to determine the candidate sets for coarsening. They store them in list  $\mathbf{C}$  and their indices in list  $\mathbf{I}^{\mathbf{C}}$  ( $\mathbf{I}_j^{\mathbf{C}} = \{i | v_i \in \mathbf{C}_j\}$ ). Depending on cost functions  $c^d$  and  $c^q$ , the algorithm decides which sets to coarsen.  $c^d$  describes the similarity of two vertices, i.e., the distance between their feature vectors. This ensures that nodes with similar features will be aggregated.  $c^q$  describes the similarity between the vertex labels, again giving aggregation priority to nodes with similar labels. The authors combine these two measures by using the logarithm of the geometric mean:

$$r_i = \frac{c_i^d + c_i^q}{2}.$$

The best-scoring candidates are considered for further evaluation. The number of candidates depends on the value of parameter  $\alpha$ . The algorithm stores all selected candidates into list  $\mathbf{C}_L$ . Once the list is full, the algorithm forms super nodes and stores them into list  $\mathbf{S}$  and their indices into list  $\mathbf{I}^{\mathbf{S}}$ . Before the super vertex is formed, the algorithm weighs each node representation and decides which vertex is the most representative of each candidate set. The feature vector of the super vertex is calculated accordingly. The representative vertex is denoted as  $\bar{\mathbf{S}}_i$ . Lastly, edges are added between super nodes if a single edge existed between the candidate sets' nodes. The above outlines one pass of Algorithm 2.

### 3 Results

This section presents the results of all graph reduction methods applied to the KnowledgeBase graph. The results are in two forms. The figures show the reduced graphs i.e. the outputs of the reduction methods. Table 2 shows the properties and reduction rates of the resulting graphs.

Algorithm	$ V $	$ E $	Reduction rate $ V $	Reduction rate $ E $
Original	25	39	0.00%	0.00%
Cytocoarsening	30	88	20.00%	125.64%
Graph-coarsening-1.1	10	13	-60.00%	-66.67%
Symmetry compression	17	12	-32.00%	-69.23%
GraphPRI-0-0	19	17	-24.00%	-56.41%
GraphPRI-0-2	16	27	-36.00%	-30.77%
GraphPRI-0-3	17	29	-32.00%	-25.64%
GraphPRI-0-4	14	25	-44.00%	-35.90%
GraphPRI-0-5	30	66	20.00%	69.23%
GraphPRI-0-1000	32	73	28.00%	87.18%
GraphPRI-0.001-2	27	60	8.00%	53.85%
GraphPRI-0.001-1000	32	74	28.00%	89.74%

Table 2: The reduction rates of the resulting graphs returned by all tested algorithms. The numbers next to the GraphPRI algorithm are the values of its parameters  $\alpha$  and  $\beta$ .

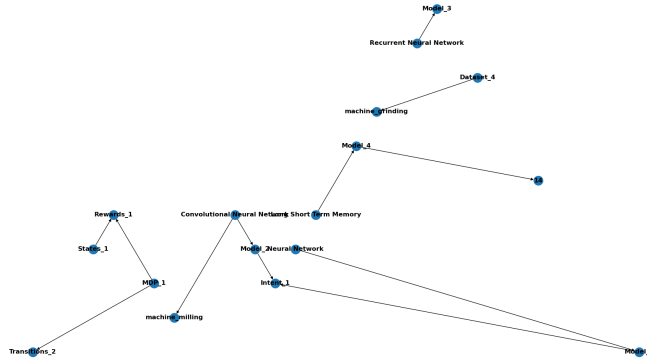


Figure 3: The residual graph of KnowledgeBase after removing all redundant edges have been removed.

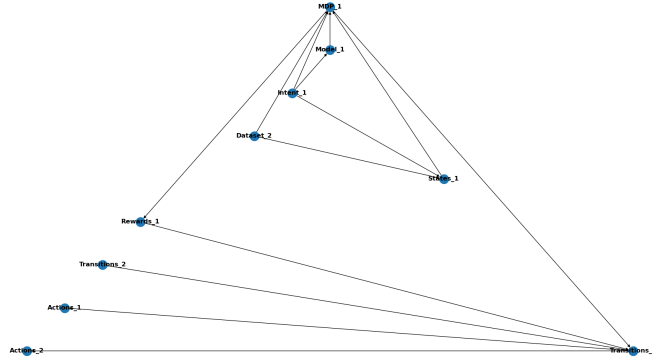


Figure 4: The result of reducing the graph with the graph-coarsening-1.1 tool.

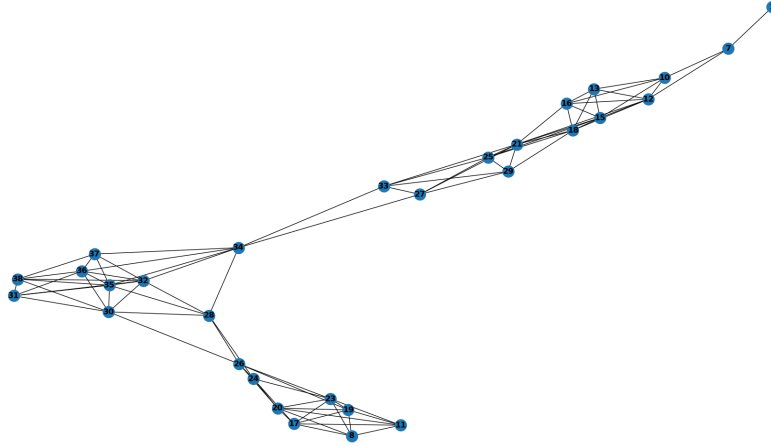


Figure 5: The result of reducing the graph with the cytoacoarsening tool.



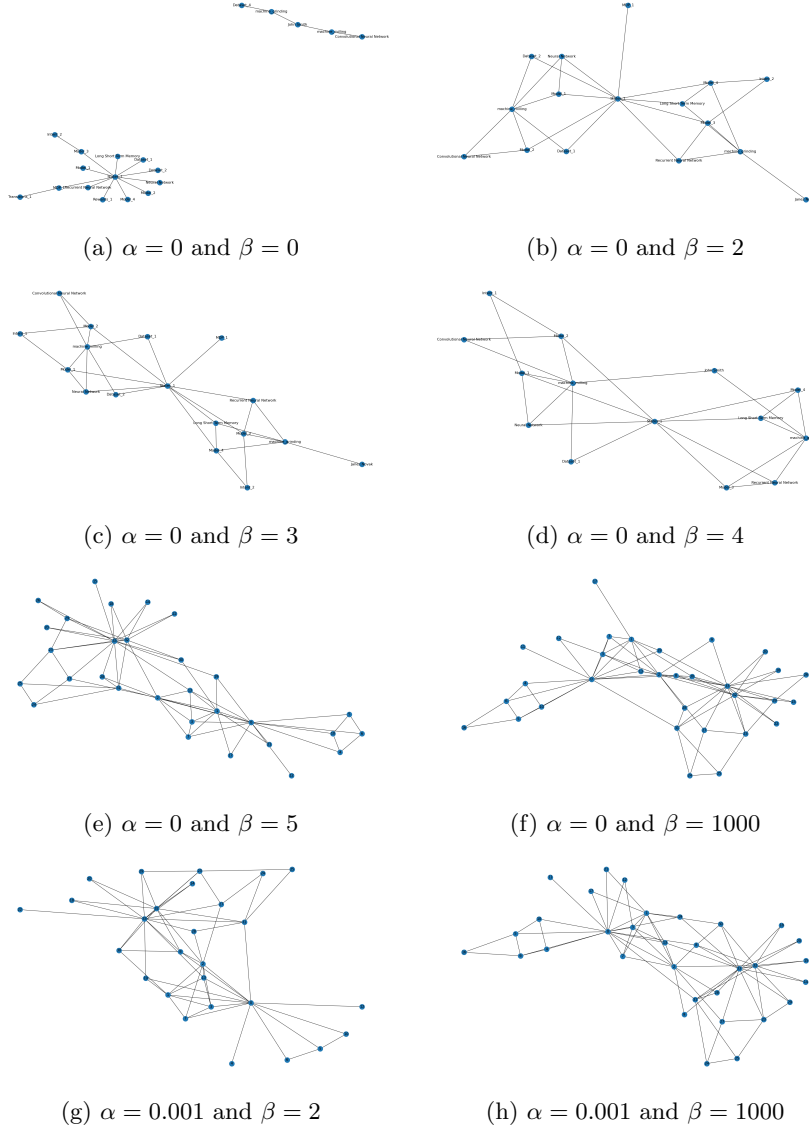


Figure 6: The graphs resulting from Graph PRI method given the parameters listed below the image.

## 4 Conclusion

In this report, we explored the use of graph reduction methods on graphs stemming from complex decision-making systems. The main goal of this method is to reduce the size of the graph while keeping all its relevant properties. This

should combat the large size of the graphs that stem from the complex systems. We tested one graph compression method, two graph coarsening methods, and one graph sparsification method. We compared the properties and reduction ratios of all resulting graphs.

Compression with graph symmetries reduced the graph to three components, thus making it likely useless in real scenarios. Cytocoarsening, in this case, inflated the graph because we did not have the vectorized version of the data table. This method would yield a higher quality reduction if the table were correctly vectorized. Graph reduction with spectral and cut guarantees yields a reduction that could be useful, providing it is not too harsh. GraphPRI yielded the two most promising reductions when its parameters were  $(\alpha = 0, \beta = 2)$  and  $(\alpha = 0, \beta = 3)$ .

For future work, we would have to test the resulting graphs to see how well they generate results in the decision-making process. It would be prudent to test other graph reduction frameworks in this domain, such as graph neural network-based ones.

## References

- [1] Čibej, Uroš and Mihelič, Jurij. "Graph automorphisms for compression," Open Computer Science, vol. 11, no. 1, (2021), pp. 51-59. <https://doi.org/10.1515/comp-2020-0186>
- [2] Kuhar, Yannick, and Uroš Čibej. "Compressing directed graphs with local symmetries." SWORDS 2023 (2023): 38.
- [3] Loukas, Andreas. "Graph reduction with spectral and cut guarantees." Journal of Machine Learning Research 20.116 (2019): 1-42.
- [4] Yu, Shujian, et al. "Principle of relevant information for graph sparsification." Uncertainty in Artificial Intelligence. PMLR, 2022.
- [5] Chen, Chi-Jane, Emma Crawford, and Natalie Stanley. "A Graph Coarsening Algorithm for Compressing Representations of Single-Cell Data with Clinical or Experimental Attributes." PACIFIC SYMPOSIUM ON BIO-COMPUTING 2023: Kohala Coast, Hawaii, USA, 3–7 January 2023. 2022.
- [6] ExtremeXP. <https://extremexp.eu/> [Online, Accessed: June 3, 2024] 2024.