A Scenario Generation Method: Firebreak Placement using Graph Variational Autoencoders and Stochastic Optimization

Lucas MURRAY HIDALGO*
lucasmurrayh@gmail.com
Ecole Normale Supérieure, Paris-Saclay

Antoine SICARD*
antoine.sicard96@gmail.com
Ecole Normale Supérieure, Paris-Saclay

1 Abstract

The severity and frequency of wildfires are increasing worldwide, driven by global warming, causing devastating effects on human activities and landscapes. In order to limit these negative effects, a strategy consists in placing firebreaks at judicious places. To address this, Vilches et al. 2023 developed a two-stage stochastic optimization approach applied on fire spread simulation data. However, their approach suffers from numerical tractability when increasing the number of scenarios and lack of robustness when evaluated outside the scenarios considered for the solution computation.

In this work, instead of giving random fire spread simulations to the optimizer, we seek to select a relevant subset of scenarios. To do so, we used two Graph Variational Autoencoder approaches to encode each simulated graph of fire spread into a low-dimensional latent space. By doing clustering in the latent space, we identified better representative simulations to give to the optimizer. Our results showed a gain in optimizing performance between 3.5 and 6 % compared to the initial strategy used in the article. This preliminary study paves the way for deeper research and further refinement of the developed approach.

2 Preface

Our work is mainly based on three scientific articles: Vilches et al. 2023, Simonovsky and Komodakis 2018 and Kipf and Welling 2016. Our objective is to improve the method proposed by Vilches et al. 2023 by implementing two versions of Graph variational autoencoders: Graph VAE (Simonovsky and Komodakis 2018) and VGAE (Kipf and Welling 2016). Specifically, we aim at exploring ways of exploiting the information and properties from a latent space generated through VAEs to enhance optimization methods.

3 Introduction

Recent scientific data show that the devastating effects of wildfires are going to increase in the next decades, as their frequency and intensity grow with global warming (Jones et al. 2024; Cunningham, Williamson, and Bowman 2024; Burton et al. 2024). This is illustrated by several examples of massive wildfires in America (such as California or Canada) in the past few years (Jones et al. 2024). These drastic wildfires have significant consequences, affecting both human activities and biodiversity (Park et al. 2024; Balmes and Holm 2023; Grant and Runkle 2022; Gajendiran, Kandasamy, and Narayanan 2024).

It is crucial to implement strategies to reduce such negative outcomes. One promising approach is to use firebeaks (such as firebreak trenches, fire-resistant buildings, or cleared vegetation strips), which can effectively stop or slow down the progression of wildfires, and reduce their destructive potential (Agee et al. 2000; Carrasco et al. 2023). However, the implementation of such firebreaks can be costly and may affect negatively the landscape. Therefore, it is essential to place them judiciously by seeking a trade-off between minimizing the spread of wildfires and limiting the associated environmental and economic constraints.

As a consequence, Vilches et al. 2023 developed a two-stage stochastic optimization approach, solved via mixed-integer linear programming to address firebreak strategic placement. For a given forest, they simulated fire propagation by a graph of nodes and edges generated by the *Cell2Fire* software (Pais et al. 2019). For each simulation, they counted the number of graph nodes, representing the number of burned cells in the forest. The first optimization stage allocates firebreaks in cells of the landscape and the second stage evaluates the performance of each firebreak placement for each simulated scenario, by calculating the number of burned cells. Once an optimal firebreak allocation is determined, it is then evaluated by running new fire spread simulations in the treated forest, now including the optimal firebreaks. This approach shows reductions in burned areas compared to random firebreak placements, demonstrating its effectiveness in wildfire management. Notably, stochastic optimization in forestry is rare, with heuristic methods dominating. Exceptions also include Kabli, Gan, and Ntaimo 2015 and González-Olabarria and Pukkala 2011. Among these, Vilches et al. 2023 stands out as one of the first stochastic optimization approaches to use spatially explicit scenarios.

However, while the approach proposed by Vilches et al. 2023 is effective at reducing fire spread, it is hindered by its complexity and low numerical tractability, which is highly sensitive to the number of scenarios incorporated into the model. This in time represents a trade-off between the benefits of incorporating additional scenarios and the associated computational expenses (see article). In addition, stochastic optimization can be very sensitive to which scenarios are included, leading to a lack of robustness; this is evidenced by their solutions performing significantly worse when tested on out-of-sample scenarios, highlighting the critical need for representative scenario selection (see also Ahmed 2013). To overcome the limitations, Vilches et al. 2023 highlight the need to develop new algorithms able to cleverly select an appropriate subset of simulated scenarios to give to the optimizer (as explored in Ahmed 2013).

4 The Two-Stage Optimization Model

In this section we introduce more deeply the model developed in Vilches et al. 2023, which is the basis of our work. As can be inferred from it's name, this model divides the firebreak placement problem into two stages. The first stage is to decide the firebreak placement, i.e. determining which set of cells will be treated, subject to α, which

*Both authors contributed equally to this research.

represents the percentage of land to be used as firebreak selected by the decision-maker (in its simplest formulation). Thus, they introduce a binary variable y_v that takes the value 1 if a firebreak is placed at the node $v \in \mathcal{N}$ and 0 otherwise. Meanwhile, in the second stage, the performance of these firebreak placements is evaluated in each scenario. Specifically, given the location of the firebreak determined in the first stage, the second stage calculates the number of burned cells in each scenario, taking into account the spread of wildfire simulated. Thus, let x_v^s be a binary variable that takes value 1 if node $v \in \mathcal{N}^s$ is burned in the scenario s after the firebreak placement and 0 otherwise. So the value of $L^s(y, w, \tau_s)$ can be calculated as $\sum_{v \in \mathcal{N}} w_s x_v^s$. The objective is the optimization of the expected value (EV) of the burned cells. Thus, it follows that the mathematical model can be expressed in the following manner:

$$\underset{x,y}{\text{minimize}} \quad \sum_{s \in \mathcal{S}} \rho^s \sum_{v \in \mathcal{V}} w_v x_v^s \tag{1a}$$

$$\text{subject to:} \sum_{v \in \mathcal{N}} y_v \leq \alpha |\mathcal{N}|, \tag{1b}$$

$$x^s_{v(s)} = 1 \qquad s \in \mathcal{S}, \tag{1c}$$

$$x_v^s \leq x_u^s + y_u \qquad v \in \mathcal{N},\ u \in \delta_+^s(v)\ s \in \mathcal{S}, \tag{1d}$$

$$\mathbf{x}, \mathbf{y} \in \{\mathbf{0}, \mathbf{1}\}. \tag{1e}$$

Equation (1a) minimizes the expected value of weighted burned nodes. This term corresponds to the sum of all burned cells v weighted by parameter w_v over all scenarios with a certain probability of occurrence ρ. Constraints (1b) define the limit of cells used as a firebreak based on the value of α. Constraints (1c) establish the origin of the wildfire, and constraints (1d) define the propagation of fire in each scenario. This is as follows: given a scenario s if the node (or cell) v is burned, an incident node u from node v must also be burned unless a firebreak is allocated. Thus, given a starting ignition node, $v(s)$ in each scenario, the firebreaks limit the wildfire spread, and a decision has to be made in every node in which a firebreak can be assigned to stop the fire such that minimizes the objective function. Finally, the constraint (1e) defines the binary variables $\mathbf{x}$ and $\mathbf{y}$ declaring that not half harvest can be done at each node. Note that if the firebreak intensity is equal to zero (i.e., $\alpha = 0$, no firebreaks), the constraints (1d) recover the fire scar of each scenario.

5 Developed approach and results

5.1 Motivation and overview of the proposed approach

As mentioned in Section 3, the primary motivation for this work is to address specific limitations identified in Vilches et al. 2023. In particular, we seek to improve the robustness of the solutions generated by the optimization model, which tend to perform significantly worse in simulated scenarios not considered during the computation of the solution compared to those that were included, and their improvement with respect to random solutions is considerably reduced. To tackle this, we propose a novel method for selecting scenarios to input into the two-stage optimization model, ensuring they are representative of the full spectrum of possible scenarios. This approach offers two main advantages: (1) it generates solutions that capture the general trends in fire-spreading behavior for a given landscape, and (2) it has the potential to reduce the computational demands of the optimization model by achieving equivalent results to randomly selected scenarios but with fewer samples.

A logical approach to selecting representative scenarios is clustering. However, many clustering techniques rely on computing distances between the objects to be grouped — a process that is far from straightforward for graphs. Classical methods often utilize graph kernels (Kriege, Johansson, and Morris 2020), but recent advancements in Generative Artificial Intelligence provide a compelling alternative. Beyond the general appeal of exploring AI-based solutions, generative modeling presents a specific advantage for clustering tasks: most clustering algorithms produce centroids that do not correspond to in-sample points. This limitation necessitates either (1) selecting the nearest in-sample point to the centroid or (2) devising a mechanism to translate points from the latent space back to the original space. Generative modeling naturally facilitates the latter.

In this work, we leverage two versions of one of the most classical graph generative models: Graph Variational Autoencoders. This choice is motivated by their balance of expressive power and simplicity, making them an appropriate starting point for the proposed application. The workflow of the proposed approach can be summarized as follows:

(1) Train a Graph Variational Autoencoder to encode graphs into a latent representation.
(2) Map the dataset into the latent space using the trained model.
(3) Perform clustering in the latent space.
(4) Select representative scenarios based on the clustering results.
(5) Solve the two-stage optimization model using the selected scenarios as input.

5.2 Description of simulated fire spreads

We selected a relatively simple forest model (*Sub20*, as described in Vilches et al. 2023) and Matias Vilches from the Fire Management & Advanced Analytics Center in Chile, generated 50,000 fire-spread graphs for us using the *Cell2Fire* software (Pais et al. 2019). In these graphs, each node represents a burned forest cell, and two nodes are connected if the fire spread from one to the other. Vilches et al. 2023 did not explicitly describe all the simulations they generated. We identified this as a potential limitation of their study, as understanding the distribution of simulations and identifying any potential outliers or aberrant cases is crucial before conducting further analysis, particularly if we want to reproduce the data. Besides this, we need to analyze the properties of the graphs in the dataset to evaluate the nature of the reconstructions generated by our models.

We analyzed each simulation by counting the number of burned cells (i.e., nodes in the generated graph). We observed that most fires spread across 275 burned cells, with a mean of 210 (Figure 1A). Interestingly, some fires did not spread at all, while others resulted in intermediate levels of burning. Consistent with the assumption that a node can only burn once, we did not detect any cycles in the

generated graphs. Furthermore, all graphs were confirmed to be Directed Acyclic Graphs (DAGs) and weakly connected (only contain one connected component). Most nodes had a single incoming connection, while a smaller proportion had 2 or 3, which occurred when fire reached the same cell simultaneously from multiple directions. Nodes with no incoming connection represent ignition points (Figure 1B). Some nodes have no outgoing connections (the fire doesn't spread further), and only a few nodes exhibited more than 3 outgoing connections (Figure 1C).

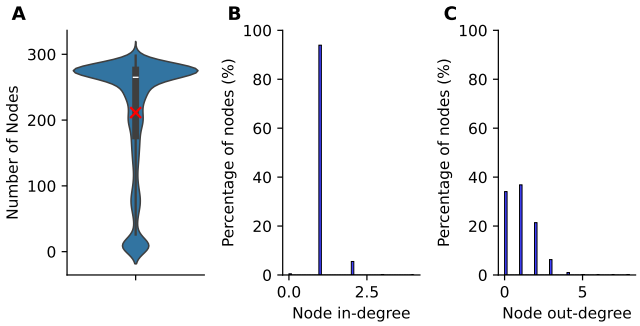

Figure 1: Description of all simulated fire graphs. (A) Distribution of number of nodes per simulated graph (violin plot). Red cross, mean; white bar, median. **(B-C)** Distribution (%) of node degree for all graphs, showing in-degree (number of incoming connections, **A**) and out-degree (number of outgoing connections, **B**)

5.3 Latent Space Generation

As mentioned briefly in 5.1, for the construction of a latent space, two versions of the Graph Variational Autoencoders were used, this because of the inherent limitations of each.

5.3.1 VGAE

The Variational Graph Autoencoders model proposed in Kipf and Welling 2016 addresses the task of unsupervised representation learning for graphs, emphasizing tasks like link prediction. The model assumes a graph $G = (V, E)$ with N nodes, an adjacency matrix A, and node features X. For this work, X is taken to be a vector of ones, since the main goal of the network is to capture the graph topology. Its architecture consists of two main components: a graph convolutional network (GCN) encoder and an inner product decoder.

The encoder maps all nodes into a latent matrix Z, where z_i represents the latent embedding of node i. This is achieved through:

$$q_\phi(Z|X,A) = \prod_{i=1}^{N} q_\phi(z_i|X,A), \quad q_\phi(z_i|X,A) = \mathcal{N}(\mu_i, \mathrm{diag}(\sigma_i^2)),$$

where $\mu = \mathrm{GCN}_\mu(X,A)$ and $\sigma = \mathrm{GCN}_\sigma(X,A)$ are learned through Graph Convolutional Networks.

The decoder reconstructs the graph by predicting the likelihood of edges between nodes:

$$p_\theta(A|Z) = \prod_{i=1}^{N}\prod_{j=1}^{N} p_\theta(A_{ij}|z_i,z_j), \quad p_\theta(A_{ij}=1|z_i,z_j) = \sigma(z_i^\top z_j),$$

where σ is the sigmoid function.

The objective function maximizes a variational lower bound:

$$\mathcal{L}(\phi,\theta;A,X) = \mathbb{E}_{q_\phi(Z|X,A)}\left[-\log p_\theta(A|Z)\right] + \mathrm{KL}\left(q_\phi(Z|X,A) \parallel p(Z)\right),$$

where the KL-divergence regularizes the latent space by aligning it with a Gaussian prior $p(Z)$.

In practice, $p_\theta(A|Z)$ is not computed for the whole adjacency matrix A, since it is generally big and sparse. Instead, $p_\theta(A_{ij}|z_i,z_j)$ is computed for all existent edges between the nodes and for an equal number of non-existent edges which are obtained through negative sampling. This limits the reconstructed graph to a subset of all possible nodes and therefore hinders the generality of the reconstructions. Furthermore, the latent representation for an arbitrary graph Z is a matrix of individual node representations z_i, which limits it's applicability to clustering in the latent space. We by-passed this limitation through generating a unique latent vector representation for Z by applying a pooling layer. This is far from ideal, relevant information would most likely be lost by this aggregation and, more relevant to our particular use of the architecture, we would not be able to reconstruct arbitrary points in the latent space.

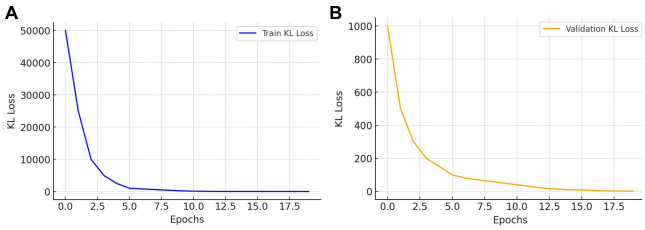

Figure 2: KL Loss for VGAE in training **(A)** and validation **(B)** sets.

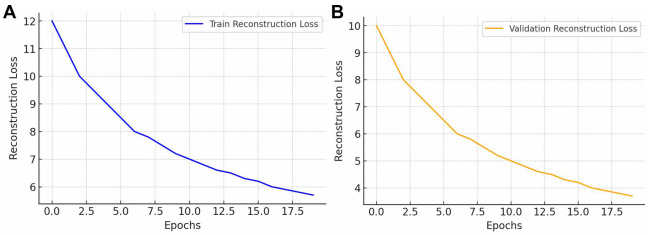

Figure 3: Reconstruction Loss for VGAE in training **(A)** and validation **(B)** sets.

5.3.2 GraphVAE

GraphVAE, proposed in Simonovsky and Komodakis 2018 is designed to generate probabilistic graphs of a fixed size k directly in a single step, sidestepping issues related to incremental or sequential graph construction. This is different from VGAE, which primarily focuses on reconstructing and predicting edges.

The encoder of GraphVAE maps an input graph $G = (V, E, F)$, defined by its adjacency matrix A, edge attributes E, and node attributes F, into a latent matrix Z using graph convolutions in

the same way as VGAE. After the generation of individual node embeddings z_i, a pooling layer is applied to the matrix Z to generate a unique vector that represents the graph z.

The decoder, in the form of a multilayer perceptron, then reconstructs a probabilistic graph $G = (\tilde{A}, \tilde{E}, \tilde{F})$ from z, where $\tilde{A}$ represents the adjacency matrix, $\tilde{E}$ the edge attributes, and $\tilde{F}$ the node attributes. The reconstruction is performed probabilistically:

$$p_\theta(G|Z) = p_\theta(\tilde{A}|z)p_\theta(\tilde{E}|z)p_\theta(\tilde{F}|z),$$

where attributes and edges are modeled as independent random variables. In our particular case, since node and edge features are not part of the necessary output, only $\tilde{A}$ is estimated. This gives way to a similar variational lower bound as in VGAE:

$$\mathcal{L}(\phi, \theta; G) = \mathbb{E}_{q_\phi(z|G)}\left[-\log p_\theta(G|z)\right] + \mathrm{KL}\left(q_\phi(z|G) \parallel p(z)\right),$$

To align the generated graph with the input graph, GraphVAE employs approximate graph matching algorithms, such as the Hungarian algorithm (Kuhn 1955), ensuring meaningful reconstruction while preserving the probabilistic framework. Since all the graphs in the dataset are subgraphs of a global grid-like graph, for which we can impose an ordering, we bypass this matching by storing the original numbering of the node in the global graph.

In contrast to VGAE, for this approach $p(\tilde{A}|z)$ is computed exhaustively, which represents a large computational cost and is hence a big limitation of the architecture. In the original paper, results for k up to 38 nodes are presented, which is quite modest. Nevertheless, depending on the particular nature of the graphs to be encoded one might be able to reduce this limitations by exploiting on domain-specific constraints on the input graphs. For our particular application, all graphs in the dataset are subgraphs of a grid-like global graph that represents the original forest. Connections in this global graph are shaped by the geographic nature of it's source: nodes can only be connected to those associated with their neighboring cells (at most 8). Capitalizing on this fact, we were able to reduce the adjacency matrix to be estimated from 160.000 entries to 2.964, which can be effectively estimated with a reasonable number of parameters.

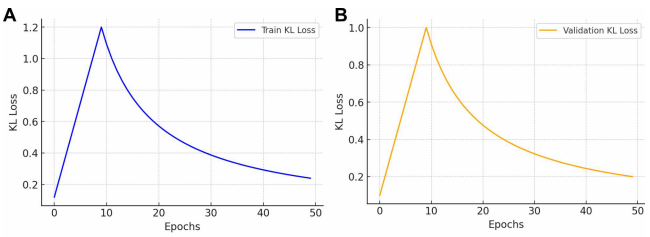

Figure 4: KL Loss for Graph VAE in training **(A)** and validation **(B)** sets.

The training and validation losses for VGAE are shown in Figures 2 and 3. Both metrics suggest that the model effectively fits the data, as evidenced by their convergence during training. Similarly, Figures 4 and 5 indicate good model fitting for Graph VAE. Notably, the KL loss initially increases during the early stages of training but subsequently stabilizes at a low value.

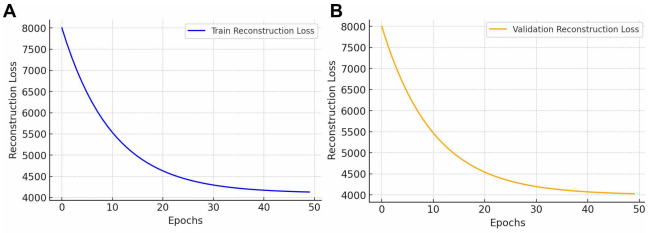

Figure 5: Reconstruction Loss for Graph VAE in training **(A)** and validation **(B)** sets.

5.4 Latent Space Analysis

To evaluate the success of the latent space generation, we proceeded using two complementary approaches. First, we analyzed the nature of the encoded graphs by comparing their reconstructions with the original graph. Second, we examined the properties of the encodings themselves, with a particular focus on addressing the curse of dimensionality. This analysis is further elaborated in Section 5.5.

We begin by assessing the reconstructions produced by VGAE.

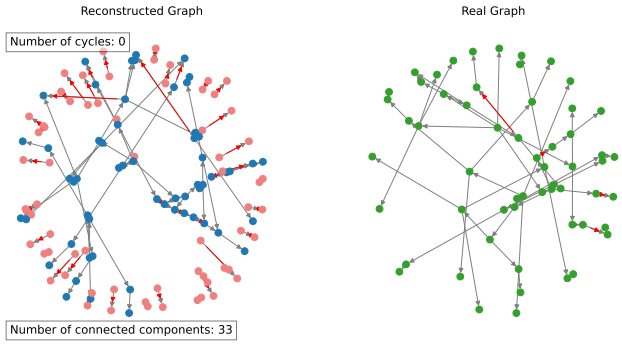

Figure 6: Reconstruction of an arbitrary graph through VGAE: On the left, the reconstructed graph from an encoding. Red edges correspond to edges not present in the original graph. Nodes with different colors represent distinct connected components. On the right, the original graph from which the encoding was generated. Red edges denote edges missing in the reconstruction.

As shown in Figure 6, the reconstructions generated by VGAE are suboptimal. While they preserve certain structural elements of the original graphs, they fail to accurately reconstruct arbitrary graphs. Specifically, VGAE effectively retains many of the edges present in the original graph but introduces fabricated edges and nodes. However, it is notable that the reconstructions produced by VGAE do not contain cycles, a critical property of directed acyclic graphs (DAGs). Additionally, the reconstructed graphs exhibit multiple connected components, which is crucial difference with the graphs in the dataset.

Figure 7 illustrates the reconstructions generated by Graph VAE, which also deviate significantly from the original graphs. Although

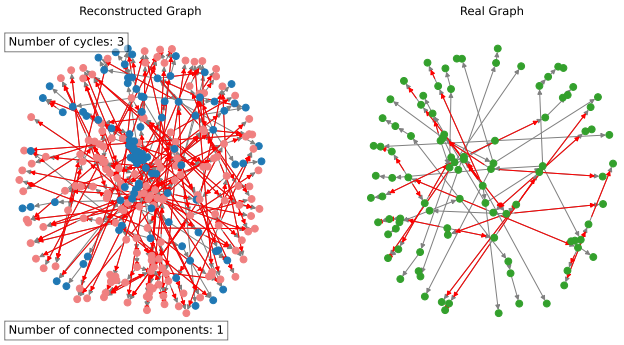

Figure 7: Reconstruction of an arbitrary graph through Graph VAE: On the left, the reconstructed graph from an encoding. Red edges correspond to edges not present in the original graph. Nodes with different colors represent distinct connected components. On the right, the original graph from which the encoding was generated. Red edges denote edges missing in the reconstruction.

these reconstructions capture some global trends, they fail to produce graphs that closely resemble the originals. Upon closer inspection, Graph VAE captures the majority of edges from the original graph but, like VGAE, introduces additional fabricated edges and nodes. Unlike VGAE, however, Graph VAE reconstructions form a single connected component, which aligns with an essential property of the studied graphs. Nevertheless, the reconstructed graphs exhibit cycles and are therefore not DAGs.

To evaluate the quality of the generated embeddings, we further analyze the properties of the approximate posterior by sampling new embeddings from it.

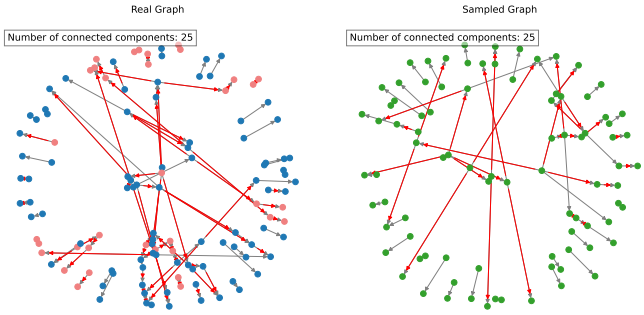

Figure 8: Sampling from VGAE's approximate posterior: On the left, the reconstructed graph from an encoding. Red edges correspond to edges not present in the graph on the right. Red nodes denote those not present in the corresponding sample. On the right, a reconstruction from a random sample drawn from the approximate posterior. Red edges represent edges absent in the original embedding's reconstruction.

As shown in Figure 8, the samples generated by VGAE preserve general structural elements of the original graph while introducing additional nodes and edges. Notably, these samples consistently maintain the number of connected components, an essential feature of the studied graphs. This property suggests that the approximate posterior effectively captures relevant topological characteristics.

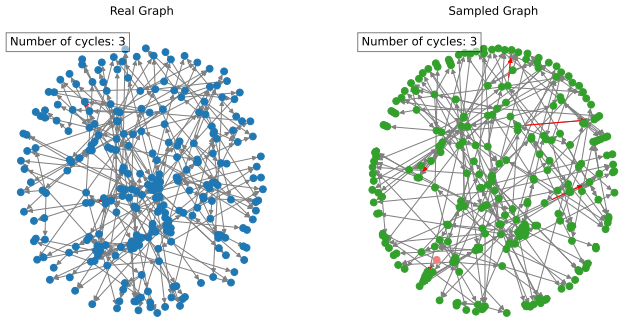

Figure 9: Sampling from Graph VAE's approximate posterior: On the left, the reconstructed graph from an encoding. Red edges correspond to edges not present in the graph on the right. Red nodes denote those not present in the corresponding sample. On the right, a reconstruction from a random sample drawn from the approximate posterior. Red edges represent edges absent in the original embedding's reconstruction.

Figure 9 illustrates that Graph VAE's samples from the approximate posterior similarly preserve the general structural elements of the graphs. A particularly noteworthy observation is that the number of cycles in the sampled graphs is consistently maintained, highlighting the ability of Graph VAE to capture this key property during sampling.

Importantly, the consistent preservation of fundamental graph properties in samples from the approximate posterior, alongside minor modifications introduced during sampling, suggests that the latent space is organized such that points in close proximity correspond to similar reconstructed graphs. This finding provides evidence that the latent space effectively encodes meaningful structural relationships between graphs (see also discussion section 6.2).

In high-dimensional spaces, the distances between points may become nearly uniform, making it challenging to identify clusters. This is an effect of the curse of dimensionality. To reduce this effect, while preserving enough latent space dimensions to adequately represent the data, we selected a 32-dimensional and a 64-dimensional latent space, for VGAE and Graph VAE respectively. This choice strikes a balance: with 32 or 64 dimensions, distances remain sufficiently distinguishable, and the impact of the curse of dimensionality should be limited, as illustrated by the simulations we did in Figure 10A.

5.5 Clustering in the Latent Space

Considering the results in the previous subsection, despite suboptimal reconstruction, we were able to represent graphs into a simpler, more compact representative latent space of 32 dimensions for VGAE and 64 dimensions for Graph VAE. In the latent space,

each graph G of the 50,000 input graphs was mapped into a multi-dimensional vector z, which is characterized by a mean $\mu_{G,d}$ and variance $\sigma_{G,d}$ for each dimension d.

Next, our goal was to apply a clustering algorithm in the latent space to identify the most representative graphs, which would then be provided to the optimizer. For this purpose, we represented each graph G in the latent space as a point with coordinates $(\mu_{G,1}, \mu_{G,2}, \ldots, \mu_{G,D})$ where $D = 32$ (for VGAE) and $D = 64$ (for Graph VAE). To ensure no single dimension dominated the distance computation during clustering, we standardized the data by scaling the μ to have a mean of 0 and a standard deviation of 1 for each dimension. Interestingly, the histogram of all pairwise distances revealed distinguishable distances, especially for the lower-dimensional latent space with the VGAE approach (Figure 10B,C), supporting the idea that clustering in these spaces made sense, with a relatively limited impact of the curse of dimensionality.

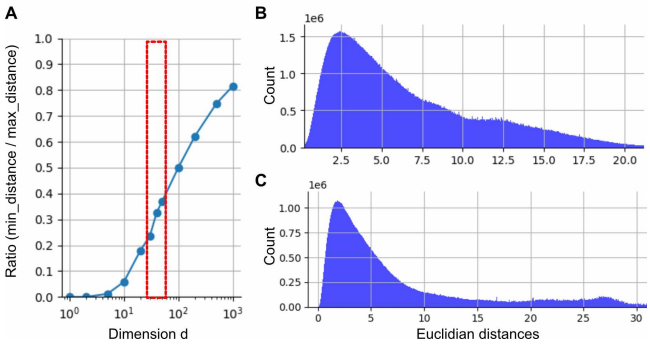

Figure 10: Dimensions and euclidian distances. (A) Impact of dimensionality on euclidian distance. For each dimension d, random 10,000 independent points uniformly distributed in $[0, 1]^d$ were simulated and the average ratio between the lowest and the highest euclidian distance was represented. The red box shows targeted dimensions for the latent spaces. **(B-C) Distribution of all pairwise distances in the latent space** for VGAE (**B**, 32 dimensions) and Graph VAE (**C**, 64 dimensions)

To explore the structure of the data in the latent space, we visualized the points in two dimensions using UMAP (Figure 11) and PCA (not shown here due to space constraints). We fine-tuned the UMAP parameters to balance the preservation of both local and global structures (see github repository for more details). We did not observe strictly defined clusters, reflecting the behavior of the autoencoder, which aimed to capture the entire data distribution rather than enforce rigidly distinct clusters. This was not a limitation, as our main goal was to select representative points — i.e., a small subset of graphs that encapsulate the overall distribution — rather than to define strictly separated clusters.

We implemented a simple clustering approach: K-Medoids (see the excellent documentation here, Algorithm 1 and Park and Jun 2009). Briefly, in its most classical form, K-Medoids is similar to K-Means, but instead of calculating virtual centroids, it selects actual data points (medoids) as cluster centers. For each cluster, it minimizes the total distance between all points and their respective medoid, iteratively updating the medoids until convergence. This choice was motivated by our first objective to identify representative points within the original dataset (as discussed in section 5.1) and the ability to predefine the number of clusters — and consequently the number of medoids — enabling precise control over how many medoids are provided to the optimizer.

Algorithm 1 K-Medoids

Require: All latent space points $L = \{x_1, x_2, \ldots, x_n\}$, number of clusters k, euclidian distance measure $d(x, y)$

Ensure: Clusters $C_1, \ldots, C_k$ and medoids $m_1, \ldots, m_k$

1: **Initialize** k medoids $m_1, m_2, \ldots, m_k$ from L (see documentation for different methods, *default : "heuristics"*)

2: **repeat**

3: **Step 1:** Assign each point $x \in L$ to the cluster of the nearest medoid:

$$C_i = \{x \in L \mid d(x, m_i) \leq d(x, m_j), \forall j \neq i\}$$

4: **Step 2:** Update the medoids m_i for each cluster C_i:

$$m_i = \arg\min_{y \in C_i} \sum_{x \in C_i} d(x, y)$$

5: **until** The medoids $m_1, m_2, \ldots, m_k$ do not change or a convergence criterion is met.

6: **return** Clusters $C_1, C_2, \ldots, C_k$ and medoids $m_1, m_2, \ldots, m_k$

We experimented with two different numbers of clusters: 20 and 100 for both VGAE and GraphVAE approaches. We projected the results in the 2-dimensional non-linear UMAP or linear PCA spaces for visualization (Figure 11A and B, due to space constraints, only the results from the 20-cluster approaches in UMAP space were represented). While the clusters were not far from each other, they provided sufficient "sampling" of the latent space to select representative medoids and, consequently to identify the corresponding representative graphs in the original space. Between the two approaches, the structure of clustering were globally similar. Most clusters were quite compact whereas some others were more spread.

5.6 Gains in optimizer performances

In the previous section, among the initial 50,000 graphs, we identified 20 or 100 medoids which are representative of the whole dataset. Then, the objective was to use them as inputs to the optimization model. To do so, we found back the corresponding graphs and we sent them to the Fire Management & Advanced Analytics Center in Chile which performed the two-stage stochastic optimization step in their model, the one developed by Vilches et al. 2023. We also sent them 20 and 100 graphs selected randomly from the whole dataset as a baseline to compare our results. After finding optimal placement of firebreaks with either medoid-corresponding graphs or random graphs, 1,000 fire spread simulations were performed on the forest with the corresponding firebreaks. When cluster-based representative simulations are given to the model, we detected between a 3.8 % and a 5.6 % reduction in wildfire spread for both VGAE and GraphVAE when compared to the random based solutions (Table 4). We found these differences to be statistically significant for

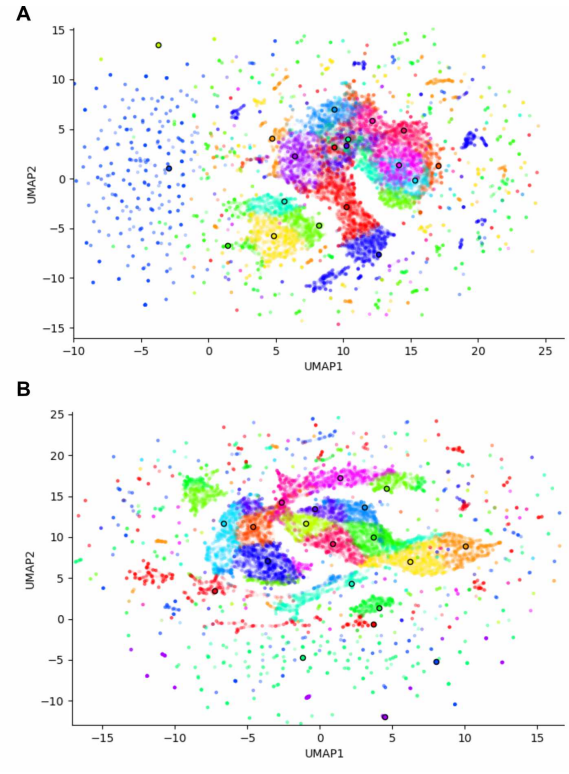

Figure 11: Clustering results for n=20. Results for the VGAE **(A)** and the GraphVAE **(B)** approaches. Clusters and data projected in the 2D-dimensional UMAP space. Clustering was performed using the K-Medoids algorithm with $n_{clusters} = 20$. Each medoid is shown as a larger point in the same color.

three of the four models, with a p-value of at least 0.05 (Tables 2 and 3).

Interestingly, at the end of the optimization step, the model evaluates the "optimal" firebreak placements on the scenarios provided as initial inputs for optimization. An initial examination suggested that, in presence of optimal firebreaks ($\alpha = 0.02$), the fire could spread more extensively in clustering-selected scenarios than in randomly selected scenarios (Table 1). Nevertheless, when looking at the percentage of burned cells without any firebreaks ($\alpha = 0$), it became evident that the scenarios selected by VGAE and Graph VAE exhibited significantly worse fire spread compared to those chosen randomly. This indicated that the scenarios selected by applying the proposed approach comprised fires of much greater magnitude and potentially higher relevance among all the scenarios. A more detailed discussion is provided in 6.3.

6 Discussion and limitations

6.1 Optimization performance

With this new approach, we successfully improved the performances of the optimizer. However, it has to be kept in mind that the increase in performance is modest and limited to ≈5 %. In this section, we discuss some limitations which may restrict the gain

Table 1: Expected Value of % cells burned for different models and number of scenarios, in sample.

Model	Scenarios	$\alpha = 0$	$\alpha = 0.02$
VGAE	20	0.5545	0.1916
Graph VAE	20	0.5391	0.1735
Random	20	0.4030	0.1935
VGAE	100	0.6069	0.2312
Graph VAE	100	0.6059	0.2315
Random	100	0.3459	0.1975

Table 2: Statistical Analysis of the differences when compared to the random scenarios solution for 20 scenarios. *, p-value<0.05.

Model	Mean	Std. Dev.	T-Statistic	p-value
Random	0.49257	0.24738	–	–
VGAE	0.46795	0.24640	-2.235	0.025*
Graph-VAE	0.47378	0.24763	-1.701	0.089

Table 3: Statistical Analysis of the differences when compared to the random scenarios solution for 100 scenarios. *, p-value<0.05

Model	Mean	Std. Dev.	T-Statistic	p-value
Random	0.48794	0.24950	–	–
VGAE	0.46028	0.24521	-2.500	0.012*
Graph-VAE	0.46244	0.24408	-2.310	0.021*

Table 4: Solution improvement with respect to the original model, *, p-value<0.05.

Scenarios	VGAE Reduction (%)	Graph-VAE Reduction (%)
20	5.00*	3.81
100	5.67*	5.23*

in performances. Additional experiments are also proposed. Interestingly, if these limitation are successfully addressed, we could expect further improvements of our results in the future.

6.2 Latent Space Construction

As showed previously, the reconstruction generated by both graph variational autoencoder methods are suboptimal. Even if this was not a major limitation to do relevant clustering in the latent space, as we successfully selected relevant medoids which improved the optimizer performance, the inability of both models to preserve critical properties of the original graphs has significant implications. Specifically, reconstructed graphs that include cycles or consist of multiple connected components are unsuitable as inputs for the optimization model. Cycles allow cells to burn multiple times, while multiple connected components permit fire to "jump" across distinct regions of the landscape — both scenarios are entirely unrealistic. This limitation prevents us from achieving one of the core objectives of this methodology: reconstructing centroids that correspond to out-of-sample points.

6.3 Clustering

As discussed in section 5.6, the fire-spreading behavior observed in the scenarios selected by the applied clustering methods corresponds to significantly larger fires compared to those obtained through random selection. This observation suggests that the clustering process successfully identified key scenarios from the original dataset of 50,000 simulated fires. Furthermore, the results presented in section 5.6 indicate that emphasizing extreme fire scenarios during the problem-solving leads to better overall solutions when evaluated across the entire dataset. However, a more detailed analysis of the properties of the selected medoids is necessary to fully validate these findings.

In this study, we employed the K-Medoids algorithm to select a representative point for each cluster directly from the latent space. Notably, K-Medoids is based on the most centrally located point in a cluster and is then less sensitive to outliers in comparison with K-Means. However, K-Medoids has several limitations. Particularly, it requires the number of clusters to be specified in advance, which necessitates careful fine-tuning of this hyperparameter. In addition, in our implementation, it only creates spherical clusters in the latent space. One way to improve the K-Medoids approach could be to test alternative distance metrics (such as Mahalanobis, cosine or Manhattan). In addition, even if K-Medoids offers a good first approach, alternative clustering methods such as HDBSCAN or Agglomerative Clustering could be considered, as they are more flexible (for an overview about different clustering methods, see sklearn documentation and Madhulatha 2012).

HDBSCAN is a hierarchical density-based method that does not require specifying the number of clusters beforehand. However, this algorithm requires a fine-tuning of its hyperparameters. Using this algorithm, we ran some preliminary experiments. We tested several hyperparameter combinations which either resulted in a large number of clusters, or only a few clusters with most points classified as noise. A next step could focus on optimizing these hyperparameters, though this may be challenging due to the poorly-separated clusters seen after projection in UMAP space.

Agglomerative Clustering is a hierarchical technique that iteratively merges points or clusters according to a chosen linkage criterion. One key advantage of this method is its ability to generate an explicit dendrogram, which allows exploration of clusters at different levels of granularity. This allows determining the optimal number of clusters (if not specified explicitly) by analyzing the dendrogram structure. However, its computational complexity of $O(n^2 \log(n))$ to $O(n^3)$ appeared to be a real issue when running multiple experiments on our local system. Despite these limitations, preliminary experiments with Agglomerative Clustering demonstrated clusters similar in shape to those produced by K-Medoids, when projected in the 2D-UMAP or PCA spaces. A next step of our approach would be to evaluate how these clusters could improve the performance of the optimizer.

Importantly, unlike K-Medoids, neither K-Means, HDBSCAN nor Agglomerative Clustering directly selects a point from the dataset as the center for each cluster. To address this, as highlighted in section 5.1, one possibility could be to select the point closest to the cluster's barycenter as a representative point. Another possibility could be to construct a new graph from the barycenter point using the decoder part of the VAE (while considering the crucial limitations discussed in section 6.2).

7 Conclusion

To conclude, we developed two approaches based on Graph Variational Autoencoders. With these techniques, we were able to encode fire spread graphs into a lower-dimensional latent space. After clustering in the latent space, we identified representative graphs which ended with better results when provided to the optimizer, compared to the classical approach used in Vilches et al. 2023. From a personal point of view, this project was really challenging as it was the first time that we used Pytorch Geometric, implemented VAEs and used clustering algorithms.

8 Acknowledgments and online resources

We would like to thank the Fire Management & Advanced Analytics Center for generating the fire spread simulations and performing the two-stage stochastic optimization. Our codebase and supplementary materials are available on **GitHub** (click to access).

References

Agee, James K et al. (2000). "The use of shaded fuelbreaks in landscape fire management". In: *Forest Ecology and Management*.

Ahmed, Shabbir (Nov. 2013). "A scenario decomposition algorithm for 0–1 stochastic programs". In: *Operations Research Letters* 41.6, pp. 565–569. ISSN: 01676377. DOI: 10.1016/j.orl.2013.07.009.

Balmes, John R. and Stephanie M. Holm (Nov. 1, 2023). "Increasing wildfire smoke from the climate crisis: Impacts on asthma and allergies". In: *Journal of Allergy and Clinical Immunology* 152.5. Publisher: Elsevier, pp. 1081–1083. ISSN: 0091-6749, 1097-6825. DOI: 10.1016/j.jaci.2023.09.008.

Burton, Chantelle et al. (Nov. 2024). "Global burned area increasingly explained by climate change". In: *Nature Climate Change* 14.11. Publisher: Nature Publishing Group, pp. 1186–1192. ISSN: 1758-6798. DOI: 10.1038/s41558-024-02140-w.

Carrasco, Jaime et al. (Sept. 15, 2023). "A firebreak placement model for optimizing biodiversity protection at landscape scale". In: *Journal of Environmental Management* 342, p. 118087. ISSN: 0301-4797. DOI: 10.1016/j.jenvman.2023.118087.

Cunningham, Calum X., Grant J. Williamson, and David M. J. S. Bowman (Aug. 2024). "Increasing frequency and intensity of the most extreme wildfires on Earth". In: *Nature Ecology & Evolution* 8.8. Publisher: Nature Publishing Group, pp. 1420–1425. ISSN: 2397-334X. DOI: 10.1038/s41559-024-02452-2.

Gajendiran, Kandasamy, Sabariswaran Kandasamy, and Mathiyazhagan Narayanan (Jan. 1, 2024). "Influences of wildfire on the forest ecosystem and climate change: A comprehensive study". In: *Environmental Research* 240, p. 117537. ISSN: 0013-9351. DOI: 10.1016/j.envres.2023.117537.

González-Olabarria, José-Ramón and Timo Pukkala (2011). "Integrating fire risk considerations in landscape-level forest planning". In: *Forest Ecology and Management* 261.2, pp. 278–287. ISSN: 0378-1127. DOI: https://doi.org/10.1016/j.foreco.2010.10.017. URL: https://www.sciencedirect.com/science/article/pii/S0378112710006298.

Grant, Emily and Jennifer D. Runkle (May 1, 2022). "Long-term health effects of wildfire exposure: A scoping review". In: *The Journal of Climate Change and Health* 6, p. 100110. ISSN: 2667-2782. DOI: 10.1016/j.joclim.2021.100110.

Jones, Matthew W. et al. (Aug. 14, 2024). "State of Wildfires 2023–2024". In: *Earth System Science Data* 16.8. Publisher: Copernicus GmbH, pp. 3601–3685. ISSN: 1866-3508. DOI: 10.5194/essd-16-3601-2024.

Kabli, Mohannad, Jianbang Gan, and Lewis Ntaimo (2015). "A Stochastic Programming Model for Fuel Treatment Management". In: *Forests* 6.6, pp. 2148–2162. ISSN: 1999-4907. DOI: 10.3390/f6062148. URL: https://www.mdpi.com/1999-4907/6/6/2148.

Kipf, Thomas N. and Max Welling (2016). *Variational Graph Auto-Encoders*. arXiv: 1611.07308 [stat.ML].

Kriege, Nils M., Fredrik D. Johansson, and Christopher Morris (Jan. 2020). "A survey on graph kernels". In: *Applied Network Science* 5.1. ISSN: 2364-8228. DOI: 10.1007/s41109-019-0195-3.

Kuhn, H. W. (1955). "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2, pp. 83–97. DOI: 10.1002/nav.3800020109. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109.

Madhulatha, T. Soni (Apr. 2012). "AN OVERVIEW ON CLUSTERING METHODS". In: *IOSR Journal of Engineering* 02.4, pp. 719–725. ISSN: 22788719, 22503021. DOI: 10.9790/3021-0204719725.

Pais, Cristobal et al. (2019). *Cell2Fire: A Cell Based Forest Fire Growth Model*. arXiv: 1905.09317 [stat.CO].

Park, Chae Yeon et al. (Nov. 2024). "Attributing human mortality from fire PM2.5 to climate change". In: *Nature Climate Change* 14.11. Publisher: Nature Publishing Group, pp. 1193–1200. ISSN: 1758-6798. DOI: 10.1038/s41558-024-02149-1.

Park, Hae-Sang and Chi-Hyuck Jun (Mar. 2009). "A simple and fast algorithm for K-medoids clustering". In: *Expert Systems with Applications* 36.2, pp. 3336–3341. ISSN: 09574174. DOI: 10.1016/j.eswa.2008.01.039.

Simonovsky, Martin and Nikos Komodakis (Feb. 9, 2018). *GraphVAE: Towards Generation of Small Graphs Using Variational Autoencoders*. arXiv: 1802.03480[cs].

Vilches, Matías et al. (Nov. 24, 2023). *A two-stage stochastic programming approach incorporating spatially-explicit fire scenarios for optimal firebreak placement.*