Mobility Network Graph Generation A Bio-Inspired approach

Amir Gheser, Matteo Mascherin, Mattia Rigon

***Abstract*—The increasingly crucial role of human displacements in complex societal phenomena, such as traffic congestion, segregation, and the diffusion of epidemics, is attracting the interest of scientists from several disciplines. In this work, we address mobility network generation, i.e., generating a city's entire mobility network, a weighted directed graph in which nodes are geographic locations and weighted edges represent people's movements between those locations, thus describing the entire mobility set flows within a city. Our solution is a graph evolution algorithm, which uses a Bio-Inspired algorithm to evolve an initial population consisting of real mobility networks. By relying on NSGA II, our approach is able to tackle graph generation by treating the task as a multi-objective optimization problem. Our method achieves results similar to other neural models, which are harder to train and less transparent.**

I. Introduction

THE motivations that spark interest towards deepening and augmenting the understanding of mobility networks are twofold: the urban environment has undergone an increasing progression in complexity, mainly caused by human displacements such as the spread of epidemics like COVID-19; the arising need to address detrimental phenomena which cause collective harm such as traffic congestion, air pollution, segregation, and epidemic spread. Consequently, predicting and simulating people's movements in an urban environment is crucial for enforcing welfare measures. To tackle said issue, generating plausible mobility flows, which are flows of people among a set of geographic locations given their demographic and geographic characteristics (e.g., population and distance), is the focus of our work.

Conventionally, such networks have been generated according to the Gravity model and Radiation model. However, in recent literature Mauro et al. [1] propose a GAN-based architecture that generates synthetic networks that approximate real test networks accurately under several evaluation metrics. To this regard, our objective is to study the effectiveness in generation of such networks by exploiting a SOTA graph evolution genetic algorithm [2].

II. Method

Following Leither et al. [2], we adopted a genetic algorithm to evolve graphs with specific properties by designating the desired properties as objectives. NSGA-II [3] optimizes conflicting objectives by preserving solutions along the Pareto front of the conflicting objectives and maintains diversity among the solutions by rewarding the ones that occupy divergent regions of the fitness landscape. With this approach graphs that satisfy user-specified properties, to the extent possible, are found, whilst also producing diverse populations of satisfactory graphs.

A. Individual Representation

A mobility network can be simply represented using a directed graph. In order to represent a graph we make use of the adjacency matrix, which is able to capture all the features of the graph. This will be relevant when we apply crossover and mutation operators implemented by Leither et al. [2], where other representation methods such as adjacency lists would be harder to handle because they would require to redesign the crossover and mutation operations. Fundamentally, each graph will be encoded using an adjacency matrix M, in which edges are stored for each node (i, j) such that:

$$M_{i,j} = w_{i,j}, \quad w_{i,j} \in [w_{min}, w_{max}]$$

where $w_{i,j}$ represents the flow (if not zero) between two nodes (2 geographical positions). One of the ways the genetic algorithm creates variation is by mutating the values in the adjacency matrix (see Section II-E). However, while changing the edge-weights via standard mutations is easy, removing edges is improbable because the weight must become exactly zero for the edge to be removed. To overcome this limitation, Leither et al. [2] represent each graph with an indirectly encoded adjacency matrix, which we refer to as the graph's genotype. The genotype of every individual is another matrix M^G such that:

$$M^G_{i,j} = \text{sparsify}(M), \quad w^G_{i,j} \in [0, 1]$$

. By using a sparsity function more values in a certain range will be prone to be set either to w_{min} (0 in our case) or to w_{max}. The sparsity function is a mapping function which maps values from $[0, 1] \rightarrow [w_{min}, w_{max}]$. An example of such graph encoding can be seen in Figure 2.

B. Initial Population

In the official implementation of graph evolution[2], the initial population is generated in a completely random way. In our case, we want to evolve different populations, starting from one of the datasets used in [1]: four public mobility datasets, describing flows of bikes and taxis in New York City and Chicago, US 1. By looking into the dataset we soon discovered that the graphs are quite sparse and that is why we decided to initialize the population by using directly real individuals. To introduce some randomness in the process, every sparsity coefficient used to map the genotype to the real adjacency matrix, is sampled randomly from a uniform distribution. This

coefficient is used to describe the sparsity strength and in this way the original individual cannot be directly inserted into the population.

C. Objective Functions

The graph evolution is done to minimize an objective function error, i.e. given an objective function and a target value, the aim of the algorithm is to minimize the Mean Squared Error (MSE) of the two ($E^2 = T^2 - P^2$), where T is the target and P is the computed value. All target values are computed on the train set sampled from the population given in each dataset. Furthermore, if the computed measure is a distribution, the squared error is summed over each frequency of a value in a certain distribution. In mathematical terms:

$$freq[b^i] = freq[b^i] + 1 \quad \text{if } v_{i,j} \in [b^i_{min}, b^i_{max}] \quad \forall v_{i,j} \in M \quad (1)$$

where b^i are the discretized bins of the distribution for continuous values e.g. in the case of the flux distribution, we first normalise the flux in range $[0, 1]$ and then divide into B bins. In the same way the target bins are computed like so:

$$\hat{b}^i = \frac{1}{N} \sum_{j=0}^{|\mathcal{D}_{train}|} b^{i,j}$$

where $b^{i,j}$ is the i^{th} bin with values ranging in $[b^i_{min}, b^i_{max}]$ and coming from every individual in the population $\mathcal{D}_{train}$.

In order to get a better understanding of the performance achieved by the algorithm we decided to follow some objective functions focusing on *local* properties (single node analysis) and *global* properties of the graph (group of nodes analysis). We evaluated different subsets of the following objective functions:

1) *In-Degree*: The distribution of in-degrees of all the nodes in a graph.
2) *Out-Degree*: The distribution of out-degrees of all nodes in a graph.
3) *Degree*: The distribution of the sum of in-degrees and out-degrees of all nodes in a graph.
4) *Flux*: We introduced the flux metric as the distribution of normalized edge weights in the graph. At different times the mobility in cities varies in intensity but the distribution of the flux is more or less the same i.e. there will be some main roads, with a certain frequency, which in proportion have more flux than other roads.
5) *Topology*: It measures how "close" the neighbors of a node are to being a complete subgraph (i.e., how interconnected the neighbors are). It gives a sense of the local density of a graph.

D. Measuring Diversity

The original NSGA-II algorithm employs a diversity-promoting mechanism to produce a varied and representative population of solutions. However, the standard NSGA-II focuses solely on creating diversity across the objectives it aims to optimize, ensuring broad coverage of the Pareto front. By following the official implementation of Evolving Graphs[2], we extend this by also maximizing the diversity of the population in terms of all unconstrained properties. To this end, all values in the genotype matrix are incorporated into the crowding distance calculation. This adjustment encourages the population to spread out within the space of genotype matrices, rather than being confined to the space of objective trade-offs. We emphasize diversity among genotype matrices because it ensures diverse values before they are transformed by the sparsify function. A simple study on diversity is provided in the appendix A.

E. Crossover & Mutation

Following [2], there are different subtypes of mutation and crossover that may occur when an overall mutation or crossover event is triggered. The relative probability of each subtype is specified by an odds ratio of the various subtypes. There are three types of mutation operators:

1) *Point* mutation
2) *Offset* mutation
3) *Sign* mutation

A point mutation replaces a randomly selected position in the genotype matrix with a new random value from the range $[0, 1]$. An offset mutation adds a small random offset to a randomly selected position, and then clamps the final value to the range $[0, 1]$. A sign mutation transforms a randomly selected value from x to $1-x$, potentially converting a positive weight in the adjacency matrix to a negative weight and vice versa via the sparsify function.

There are also three types of crossover operator:

1) *Row* crossover
2) *Depth-first Traversal* crossover
3) *Breadth-first Traversal* crossover

A row crossover exchanges randomly chosen rows between two parent genotype matrices (one row encodes a single node and its outbound edges). A depth-first- and a breadth-first-traversal crossover execute a graph traversal starting from a randomly chosen node and exchange each row visited by the traversal with the same row from another matrix. The number of nodes visited and so the length of the path traversed is determined probabilistically. These graph traversal crossover operators transfer the entire traversed path, possibly preserving important graph structure and thus potentially enhancing evolvability.

F. Constraint-Handling

The NSGA-II [3] algorithm is effective in providing optimal solutions while handling both valid and invalid solutions. The solutions are always sorted so that unfeasible solutions come after feasible solutions. In order to apply the algorithm we have defined which solutions are valid or invalid. We apply statistics-based constraints both on real values and distributions of values. In the former case, we provide a target μ and standard deviation σ, obtained from the train set, and ensure that the metric calculated on the ith organism, $\hat{\mu}$ and $\hat{\sigma}$ satisfies the following inequality:

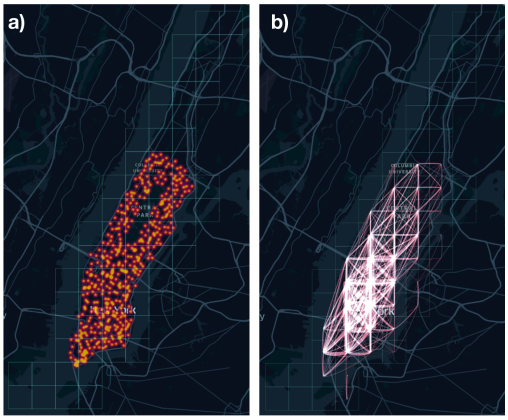

Fig. 1. Examples of a real mobility network. (a) Position of bike stations in Manhattan. (b) A daily mobility network in Manhattan, where the size of each edge is proportional to the flow they represent.

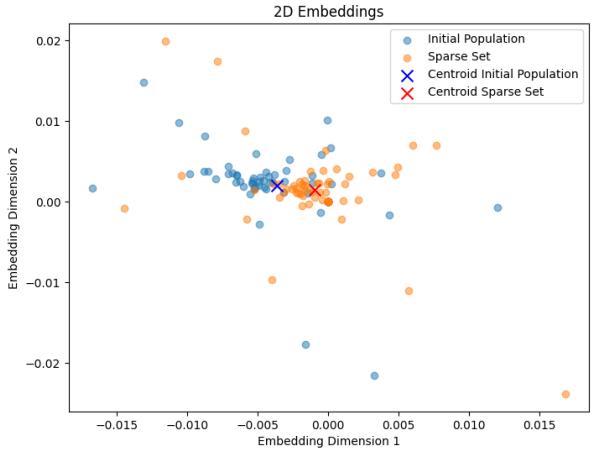

Fig. 2. Distribution of individuals sampled from the original population and from after the sparsify function is applied. Embeddings are computed using the eigen vectors of the Laplacian matrix

$$\text{IoU}\left(\mathcal{N}(\mu,\sigma),\mathcal{N}(\hat{\mu},\hat{\sigma}\right) \geq \varepsilon$$

In the latter case, given a distribution metric $dist$ for each frequency bin $dist_j$ of the distribution, we count how many bins have frequency within target mean t_j and target standard deviation $2\sigma_j$. If the count is above 80% the sample will be feasible.

$$\frac{1}{N}\sum_{j=1}^{|dist|} \mathbb{1}(v_j, t_j, \sigma_j) \geq 0.8 \tag{2}$$

where

$$\mathbb{1}(v,\mu,\sigma) = \begin{cases} 1 & \text{if } |v-\mu| <= 2\sigma \\ 0 & \text{if } |v-\mu| > 2\sigma \end{cases}$$

III. Results

In order to assess the performance of our project, we decided to focus our experiments on just one dataset from [1]. In this way, some hyperparameters [1] are fixed, while others are optimized for the particular problem. We tested the evolution on a combination of two objective functions taken from the following:

1) degree
2) topology
3) weight
4) flux

We evaluate our algorithm from different perspectives:

- Distributions of values of a certain property
- Distributions of the 2D projection of a certain population.

While the optimized properties are guided by some target values computed using a train dataset, the final comparison is done against a subset of Mobility Networks Chicago dataset. Performances achieved by a fake population generated by Leither et al.[1] are reported for comparison purposes. Firstly, we compared the KL-divergence between the real and the generated (with ours and MoGAN's method) population. The divergence is computed by comparing some key properties of every individual of a generated population, namely the *flux* and the *degree* distribution. As it can be observed in Table I, our method is able to achieved a very good result compared to MoGAN, in the degree distribution. This can be observed both in the experiments where degree was directly optimized, but also in the Topology-Flux experiment where neither the constraints nor any of the objectives focus on degree. The comparison between our method and MoGAN, shown in Figure 3, demonstrates that our approach closely replicates the properties of real graphs across multiple metrics. The flux distribution (top-right) demonstrates that our method aligns more consistently with the real distribution than MoGAN, particularly for lower normalized flux values. Similarly, the degree distribution (top-left) highlights that our method better captures the real graph's structure, whereas MoGAN exhibits a notable deviation in degree frequency. The 2D embeddings (bottom) provide further evidence that we are able to get closer to the original distribution. This can be observed by looking at the centroid of the population which is similar to the one determined by MoGAN [1] generated population. Additionally, we noticed that our generated population is a bit more sparse than the original one. Again, this is probably due to the fact that we are enforcing the diversity between individuals and adding this on top of a sparsify function, pushing the algorithm to provide a more diverse population. Despite the promising results on our metrics we emphasize the urge to formally define the metrics that define a mobility network in order to have a common ground for evaluation. The tolerance hyperparameters for evaluation are to be defined based on the problem.

IV. Limitations

We had problems identifying proper evaluation functions as they are often under-representative, and the metrics often only encapsulate one aspect of the network. Additionally, the

[1]all hyperparameter configurations can be found in our github, in the config* folders. https://github.com/MattiaRigon/BioMobilityNetworks

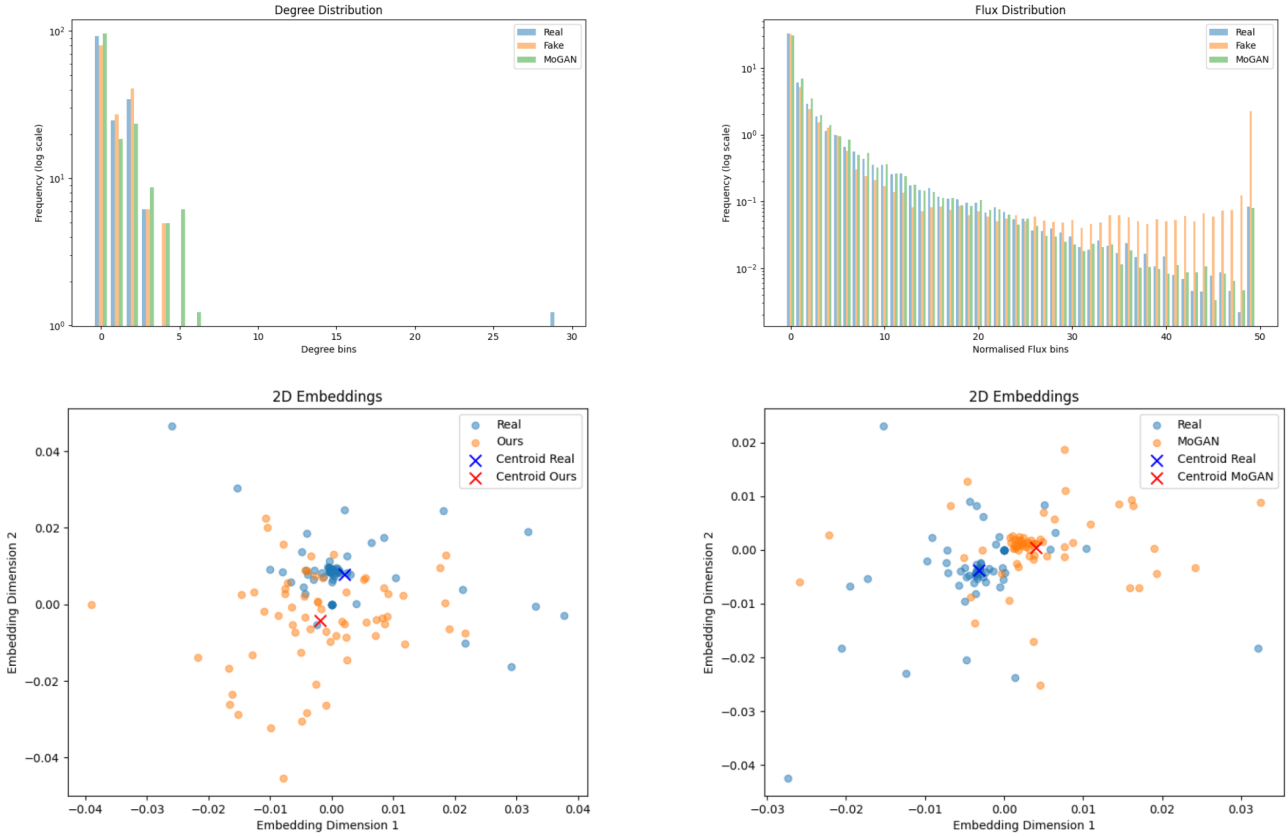

Fig. 3. Results in terms of embeddings and distributions comparison in between real, our and MoGAN generated population

Properties Optimized	**KL-divergence** against real distribution			
	Flux Distribution		**Degree distribution**	
	Ours	MoGAN	Ours	MoGAN
Topology-Flux	**0.37**	1.14	**0.21**	0.27
Degree-Topology	**0.76**	1.14	**0.23**	0.27
Degree	**0.38**	1.14	**0.21**	0.27
Degree-Flux	**0.38**	1.14	**0.20**	0.27

TABLE I
RESULTS IN TERMS OF KL-DIVERGENCE OF OUR AND MOGAN [1] METHOD ON DIFFERENT PROPERTIES DISTRIBUTIONS

error-based metrics of MoGAN [1] indicate errors but do not capture the distribution of key features such as degree and flux across the network.

V. CONCLUSION AND FUTURE WORK

In this report, we presented a novel solution able to generate graphs which adhere to some properties, starting from an initial population. By putting together the extensive work from Mauro et al. [1] and Leither et al. [2], we are able to apply a method tested just in a synthetic setting ([2]) on a real world problem using real data ([1]). Our results suggest that Bio-Inspired methods can be successfully employed to generate new samples from an initial population. Another interesting observation is that with neural models the training process is very expensive and once trained it will generate data that depends on the training set. On the other hand, our approach is capable of focusing on some properties of the original distribution, which can be useful if we need synthetic data targeting some specific features. Overall the trade-off between the two methods relies on the time (evolution is expensive) and on the hardware requirements (a GPU is not required to run our project). These are promising results that pave the way for further experiments, specifically, we would like to address some potential paths to continue the experiments such as: running all evaluations in the datasets provided by Mauro et al. [1] and including *NSGA III* [4] to work with more than 2 objective functions. Furthermore, considering the very sparse nature of the mobility networks, encoding the genome as adjacency lists could be a promising pathway for optimizing performance. While this approach is valuable without hardware acceleration, implementing a parallelized version of the algorithm could significantly improve efficiency, making it more competitive with deep learning solutions.

VI. CONTRIBUTION

All group members contribute equally to the project. The code framework to run the experiments has been developed by the whole team. For the remaining part:

- *Mattia Rigon* and *Amir Gheser* focused most on the final experiments of the project.
- *Matteo Mascherin* focused on integrating the results into the report.

The workload has been equally divided.

REFERENCES

[1] G. Mauro, M. Luca, A. Longa, B. Lepri, and L. Pappalardo, "Generating mobility networks with generative adversarial networks," *EPJ Data Science*, vol. 11, no. 1, Dec. 2022. [Online]. Available: http://dx.doi.org/10.1140/epjds/s13688-022-00372-4

[2] S. Leither, V. Ragusa, and E. Dolson, "Evolving weighted and directed graphs with constrained properties," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '24 Companion. New York, NY, USA: Association for Computing Machinery, 2024, p. 443–446. [Online]. Available: https://doi.org/10.1145/3638530.3654350

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[4] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

APPENDIX A
STUDY ON DIVERSITY

In this section we analyse the diversity across solutions obtained by optimizing on different objective functions. As outlined in Table II you can see the tradeoff between number of unique samples, evaluated on connectance, and the similarity between the metrics distributions on the fake set versus the test set. As shown the degree metrics significantly outperform all other metrics.

Optimized Properties	No. Unique Connectance	(KL) Flux Distribution	(KL) Degree Distribution
Topology-Flux	75/100	0.37	0.21
Degree-Topology	84/100	0.76	0.23
Degree	96/100	0.38	0.21
Degree-Flux	86/100	0.38	0.20

TABLE II
RESULTS IN TERMS OF CONNECTANCE WHERE CONNECTANCE IS DEFINED AS (M/N^2) GIVEN A GRAPH G WITH N NODES AND M EDGES.

In addition Figure 4 and Figure 5 show the pareto fronts of two different population, showcasing properly separated dominance levels.

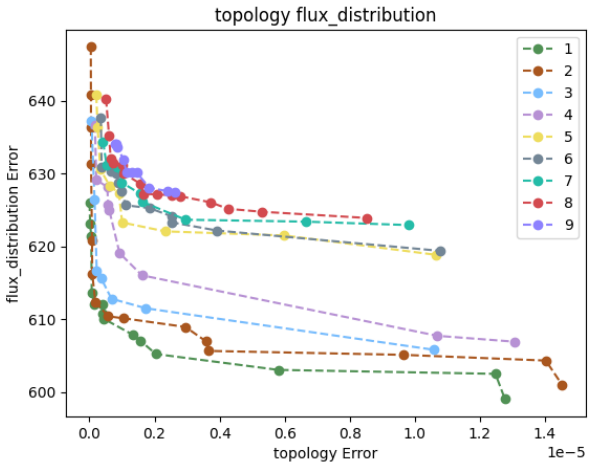

Fig. 4. Pareto topology flux

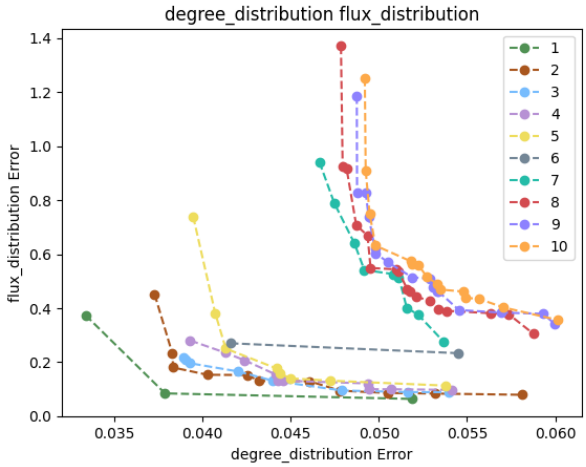

Fig. 5. Pareto degree flux