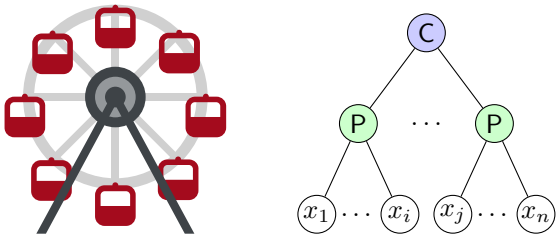
Permutation Tree Invariant Neural Architectures

Johannes Urban [1] **Sebastian Tschiatschek** [1 2] **Nils M. Kriege** [1 2]

Abstract

Exploiting symmetry as an inductive bias has become a fundamental technique in deep learning to improve generalization and sample efficiency. We investigate the design of models that are invariant to subgroups of the symmetric group defined by hierarchical structures. We propose permutation trees, which represent permutations by the ordering of their leaves and allow the reordering of siblings depending on the type of their parent, generalizing PQ-trees. We characterize the permutation trees that represent permutation groups and derive invariant neural architectures from them in a bottom-up fashion. We show that our approach learns faster with less data and achieves an improved prediction performance on a synthetic dataset.



(a) Vienna Ferris wheel (b) Permutation tree

Figure 1: Illustration of the Vienna Ferris wheel (a) and a permutation tree (b) modeling the symmetries of the participants of a ride. The passengers within the same gondola are regarded as a set, meaning that they can be permuted arbitrarily in the list of participants. However, they are grouped according to the gondolas in the order of the Ferris wheel, allowing cyclic shifts of whole groups.

1. Introduction

Symmetries naturally arise in various applications, and incorporating them into machine learning models can drastically improve their generalization and sample efficiency. For example, a molecule can be represented by a set of atoms annotated with 3D coordinates. As chemical properties depend on the spatial arrangement of atoms but not directly on their coordinate, models can benefit from invariance to rotation. Moreover, the set of atoms is typically passed to the network in an arbitrary order. However, the order should not influence the prediction, leading to the requirement of invariance to all possible permutations of the input. Such set functions arise in various applications and have been extensively investigated recently (Zaheer et al., 2017; Wagstaff et al., 2022; Amir et al., 2024). Moreover, permutation invariance is essential in methods such as graph neural networks (Xu et al., 2018; Fuchs & Veličković, 2023), where sets of neighboring nodes need to be processed. A requirement orthogonal to invariance is that of universality, i.e., in particular, there should be a parametrization of the network such that it computes a different output for all pairs of inputs with different underlying sets. Besides sets with the requirement of invariance to all possible permutations, more specific permutation groups have not been investigated to the same extent. Notable exceptions are neural architectures for sets of objects with symmetries (Maron et al., 2020) and the use of the wreath product to reflect hierarchical symmetries (Wang et al., 2020).

In this work, we investigate and propose neural architectures invariant to subgroups of the symmetric group represented by a *permutation tree*, a generalization of PQ-trees introduced by Booth & Lueker (1976) in a different context. Permutation trees represent permutations of their leaves by allowing the reordering of siblings based on the node type of their common parent. Permutation trees naturally represent permutations with a nested structure. Consider the example of a Ferris wheel illustrated in Figure 1. A function on the list of participants should be invariant to the reordering within each gondola, and these groups, in turn, should be invariant to cyclic shifts following the order of gondolas on the Ferris wheel. Formally, we aim at developing neural networks computing a function $f\colon \mathcal{X}^n \to \mathbb{R}^d$ that are invariant to a given permutation group P, i.e., for all π in P it holds

[1]Faculty of Computer Science, University of Vienna, Währinger Str. 29, 1090 Vienna, Austria [2]Research Network Data Science, University of Vienna, Kolingasse 14–16, 1090 Vienna, Austria. Correspondence to: Nils M. Kriege <nils.kriege@univie.ac.at>.

Proceedings of the Geometry-grounded Representation Learning and Generative Modeling at 41st International Conference on Machine Learning, Vienna, Austria. PMLR Vol Number, 2024.

$f(\boldsymbol{x}) = f(\pi(\boldsymbol{x}))$. Here, we restrict permutation groups to those that can be represented by permutation trees.

Our contribution. We propose permutation trees and show that they give rise to permutation groups if certain structural properties are satisfied. From such trees, we can learn functions that are invariant to the permutation group they represent by associating the input sequence with the leaves and computing embeddings in a bottom-up fashion. We derive functions to combine the embeddings of children at inner nodes depending on their type. We experimentally show that our method learns faster with less data and achieves an improved prediction performance on a synthetic Ferris wheel dataset.

2. Preliminaries

A *permutation* of a set S is a bijection $\pi\colon S \to S$. Let π be a permutation of $\{1, 2, \ldots, n\}$ and let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ a sequence of n objects in $\mathcal{X}^n$, we denote by $\pi(\boldsymbol{x})$ the sequence $(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$. Two sets of permutations P and Q of sets S and T, respectively, are *isomorphic* if there is a bijection $\varphi\colon S \to T$ such that renaming objects in P according to φ yields $P' = T$. A set of permutations P together with the composition of permutatins denoted by $\circ$ is a *permutation group* if (i) P is closed under composition, i.e., for all π, σ in P the permutation $\pi \circ \sigma$ is in P, (ii) the identity permutation $x \mapsto x$ for x in S is in P, and (iii) for all π in P the inverse π^{-1} is in P. Given a permutation group P acting on S, the *orbit* of s in S is $\{\pi(s) \mid \pi \in P\}$. The orbits of P yield an equivalence relation, and we write $x \sim_P y$ if x and y are in the same orbit.

3. Permutation Tree Invariant Networks

We introduce permutation trees to represent sets of permutations by hierarchical structures and relate them to permutation groups necessary to derive invariant networks.

3.1. Permutation Trees

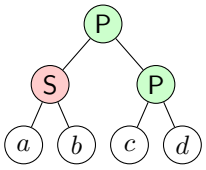
We provide a general definition of permutation trees and derive specific instances later.

Definition 3.1 (Permutation tree)**.** A *permutation tree* on a set L is a rooted ordered tree with leaves L, where each internal node v is endowed with a permutation group G_v acting on its children.

We introduce several node types associating well-known permutation groups to the internal nodes.

Definition 3.2 (Node types)**.** Let v be an internal node of a permutation tree with children $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$.

- If v is a **P-node**, then G_v is the symmetric group $\{\pi(\boldsymbol{c}) \mid \pi \in S_n\}$, allowing arbitrary reordering of the children.
- If v is a **Z-node**, then G_v is $\{\pi(\boldsymbol{c}) \mid \pi \in C_n\}$ with C_n the cycles $\pi_j\colon i \mapsto (i + j \bmod n) + 1$ for some j in $\mathbb{N}$, allowing circular shifts of the ordered children.
- If v is a **Q-node**, then G_v is $\{\boldsymbol{c}, (c_n, c_{n-1}, \ldots, c_1)\}$, i.e, the order of the children can be inverted.
- If v is a **C-node**, then G_v are the circular shifts of the children with possible inversion of their order.
- If v is a **S-node**, then $G_v = \{\boldsymbol{c}\}$, meaning that no reordering of the children is allowed.



Frontiers	Permutations
(a, b, c, d)	$\pi_1 = (1, 2, 3, 4)$
(a, b, d, c)	$\pi_2 = (1, 2, 4, 3)$
(c, d, a, b)	$\pi_3 = (3, 4, 1, 2)$
(d, c, a, b)	$\pi_4 = (3, 4, 2, 1)$

Figure 2: Permutation tree T, the frontiers of all equivalent permutation trees, and the associated permutations in one-line notation forming $\mathrm{Perm}(T)$. Note that $\pi_3 \circ \pi_2 = (4, 3, 1, 2)$ is not supported by T and, thus, $\mathrm{Perm}(T)$ is not closed under composition and not a permutation group.

Two permutation trees R and T are *equivalent*, written $R \equiv T$, if R is obtained from T by reordering siblings according to the permutations supported by their common parent. Formally, for all inner nodes v of T, there is a permutation π_v in G_v such that their application to T yields a tree isomorphic to R. The *frontier* $f(T)$ of a permutation tree T is the sequence of its leaves read from left to right. A permutation tree T on n elements *supports* the permutations

$$\mathrm{Perm}(T) = \{\pi \in S_n \mid R \equiv T \wedge \pi(f(T)) = f(R)\}.$$

Figure 2 shows an example of a permutation tree. Note that we recover the standard definition of PQ-trees introduced by Booth & Lueker (1976) when restricting permutation trees to nodes of type P and Q. The node type C is inspired by the more recent PC-trees (Wei-Kuan & Wen-Lian, 1999).

3.2. Permutation Trees and Permutation Groups

Our goal is to learn functions that are invariant to the permutations supported by a permutation tree. The conception of invariance requires that this set forms a group (Bronstein et al., 2021). However, the permutations supported by a permutation tree, in general, do not form a group; see Figure 2 for a counter-example. Although structures like PQ- and PC-trees have been used for decades and are closely related to permutation trees, to the best of our knowledge, they have not been related to permutation groups. In the following, we characterize the permutation trees that represent permutation groups.

We associate with each node v of a permutation tree T its

scope $s(v)$ containing all the leaves of the subtree rooted at v. A node v represents the permutations $\mathrm{Perm}(v, T)$ of its scope $s(v)$ defined by the subtree rooted at v. We introduce regular permutation trees and relate them to permutation groups in the following.

Definition 3.3 (Regular permutation tree)**.** A permutation tree T is *regular* if for all nodes v in T having children s, t with $s \sim_{G_v} t$ it holds $\mathrm{Perm}(s, T) \simeq \mathrm{Perm}(t, T)$.

Figure 1b shows a regular permutation tree assuming that each P-node has the same number of children. The tree depicted in Figure 2, in contrast, is not regular since the root allows its two children to be permuted, although one is an S-node and the other a P-node yielding non-isomorphic sets of permutations on their scope.

To show that regular permutation trees lead to permutation groups, we first consider a single node and its children.

Lemma 3.4. *Let v be a node in a permutation tree T with children $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$. The set $\mathrm{Perm}(v, T)$ forms a permutation group if*

(i) $\mathrm{Perm}(c_i, T)$ *forms a permutation group for all i in $\{1, 2, \ldots, n\}$,*
(ii) $\mathrm{Perm}(c_i, T) \simeq \mathrm{Perm}(c_j, T)$ *holds for all i, j with $c_i \sim_{G_v} c_j$.*

Proof. We show that conditions (i) and (ii) are sufficient to obtain a permutation group. Assume that (i) and (ii) are satisfied and $\mathrm{Perm}(v, T)$ is not closed under composition. Then there are permutations π, σ in $\mathrm{Perm}(v, T)$ with $\pi \circ \sigma$ not in $\mathrm{Perm}(v, T)$. Since π and σ are supported by the permutation tree, we obtain two permutations τ_v^π and τ_v^σ of the children $\boldsymbol{c}$ leading to π and σ, respectively. We apply the permutation $\tau_v^\pi \circ \tau_v^\sigma$ to $\boldsymbol{c}$. Since this maps only children with equivalent permutation groups, we obtain a permutation tree for $\pi \circ \sigma$ contradicting the assumption. Similarly, we can show that we obtain the identity permutation for the identity mapping of the children. The inverse of a permutation π in $\mathrm{Perm}(v, T)$ is obtained by inverting the permutation of the children, which is possible according to condition (i). We have derived the three defining properties of a permutation group, proving the result. □

We apply the result to relate regular permutation trees and permutation groups.

Theorem 3.5. *If T is a regular permutation tree, then $\mathrm{Perm}(T)$ forms a permutation group.*

Proof. We prove the result by structural induction over permutation trees. As the base case, we assume that the tree T consists of one inner node with an arbitrary number of leaves. Since the inner node v is associated with a permutation group G_v, also $\mathrm{Perm}(T) = \mathrm{Perm}(v, T)$ forms a permutation group as it is isomorphic to G_v. In the induction step, we create from a set of n trees with roots $R = \{r_1, r_2, \ldots, r_n\}$ a new tree T consisting of a new root r with children R corresponding to the roots of the n trees. The induction hypothesis allows the application of Lemma 3.4 and the result directly follows. □

3.3. Invariant Neural Networks

Let T be a regular permutation tree with n leaves. We would like to compute a function $f\colon \mathcal{X}^n \to \mathbb{R}^d$ with $f(\boldsymbol{x}) = f(\pi(\boldsymbol{x}))$ for all $\boldsymbol{x}$ in $\mathcal{X}^n$ and π in $\mathrm{Perm}(T)$. To this end, we associate the ith entry x_i of $\boldsymbol{x}$ with the ith leaf of T. We compute embeddings for the inner nodes in a bottom-up fashion from the embeddings of their children. The final output $f(\boldsymbol{x})$ is the embedding obtained for the root. The approach satisfies the invariance properties given that the embeddings $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ of the children of v are combined in a G_v-invariant way to obtain the embedding $\boldsymbol{x}_v$. Moreover, our approach is universal if the neural networks that realize the individual nodes are universal. This is the case for suitable configurations of the techniques we describe in the following for the node types of Definition 3.2.

P-nodes. We realize P-nodes using the seminal Deep Sets method introduced by Zaheer et al. (2017), i.e.,

$$\boldsymbol{x}_v = \psi\left(\sum_{i=1}^{k} \phi(\boldsymbol{x}_i)\right),$$

where the functions ψ and ϕ are realized by a multilayer perceptron.

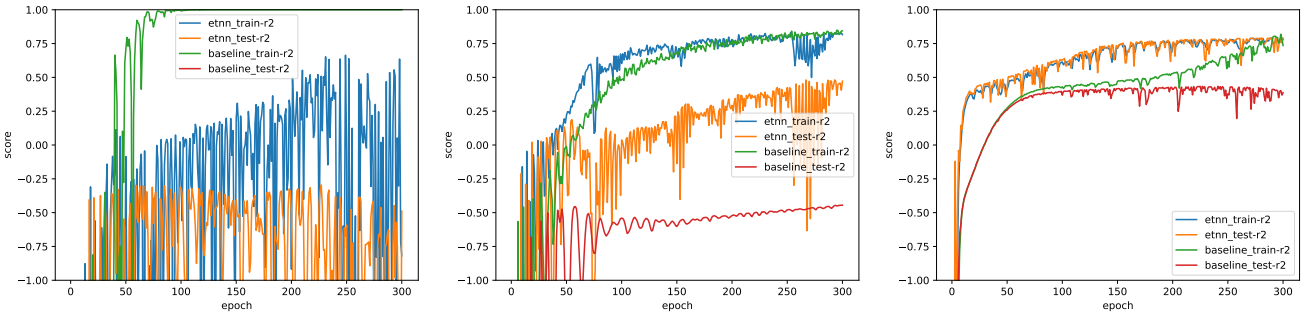
S-nodes. We realize S-nodes by concatenating the embeddings of the n children and mapping them to a fixed dimensional embedding using a neural network ϕ, i.e.,

$$\boldsymbol{x}_v = \phi(\boldsymbol{x}_1 \| \boldsymbol{x}_2 \| \cdots \| \boldsymbol{x}_k).$$

Z-nodes. We realize Z-nodes using the approach introduced by Gaiński et al. (2023) splitting the cyclic sequence of children into all possible tuples of z consecutive elements, where z in $\{2, 3, \ldots, k\}$ is a parameter. The embeddings of all tuples are regarded as sets and combined accordingly, i.e.,

$$\boldsymbol{x}_v = \psi\left(\sum_{i=0}^{k-1} \phi\left(\Big\|_{j=0}^{z-1} \boldsymbol{x}_{((i+j) \bmod z)+1}\right)\right).$$

Q- and C-nodes. We obtain embeddings for Q- and C-nodes by computing embeddings using the technique for S- and Z-nodes, respectively, for $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k)$ and $(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \ldots, \boldsymbol{x}_1)$. The two resulting embeddings are summed up and transformed using a Deep Sets like approach for sets of two elements, cf. P-nodes.



(a) $|\mathcal{D}| = 10$ (Baseline test below -1) (b) $|\mathcal{D}| = 1{,}000$ (c) $|\mathcal{D}| = 10{,}000$

Figure 3: Performance (coefficient of determination, r^2) on the Ferris wheel data set for varying amounts of data. Our approach, ETNN, achieves better performance on the test set for all amounts of available data. Refer to the main text for details.

4. Experiments

We evaluate our proposed permutation tree invariant neural architecture on synthetic data and leave the evaluation on real-world data with appropriate characteristics, e.g., in the field of stereochemistry, to future work. The code for the experiments shown in this part is stored in a GitHub repository[1].

4.1. Data

We construct a *Ferris wheel* inspired dataset in which the labels are invariant to a permutation tree with C- and P-nodes as shown in Figure 1b.

In particular, we create a data set with n samples, where each sample corresponds to an assignment of passengers to the gondolas in a Ferris wheel and an associated score. Concretely, let n_G be the number of gondolas and n_P the number of persons in each gondola. Each person is associated with person-specific data extracted from the *Sleep Health and Lifestyle Dataset* data set (Tharmalingam, 2023). The labels are computed such that the rotation of the Ferris wheel, and hence the relative position of a gondola, and the ordering of the persons within a gondola do not affect the labels. The information about the passengers is stored in arbitrary order within each gondola and such that the information about which gondola is next to each other is retained. We use varying amounts of data for training and testing the models. For a given data set size, the available data is split into a test data set (30% of the available data) and a training data set (70% of the available data). The training data is further split into 70% for the actual training and 30% for validation (mainly hyperparameter tuning). The data (but not the labels) is further normalized using a Min-Max scaler. A detailed description of the creation of the synthetic data is provided in Appendix A.

[1] https://github.com/JellyJoe13/P2_EquivariantTreeNN

4.2. Models, Training, Evaluation

We compare the following two models:

(i) BASELINE. As a baseline model we use vanilla neural networks consisting of (multiple) linear layers with ReLU activation functions followed by a linear output layer. The number of hidden layers and hidden units was chosen to ensure that the number of parameters of the baseline is close to the number of parameters of the ETNN model.
(ii) ETNN (our approach). The Equivariant Tree Neural Network (ETNN) realizes invariance to a specified permutation tree through the use of the update functions as described in Section 3.3. In general, a collection of linear layers, and non-linear activation functions is used to derive the order invariance for each internal node of the tree.

All models are trained using Adam for 300 epochs and we report the performance of the models during training on the training and the test data. Performance is measured using the coefficient of determination (r^2). Hyperparameter optimization of the number of hidden unis and layers, learning rate, and data normalization of the label (i.e., using a Min-Max-scaler or not) is performed using Optuna (Akiba et al., 2019), which runs and evaluates the experiments using the validation split of the data after 100 epochs.

4.3. Results

We report the performance of our proposed model in comparison to the baseline model in Figure 3 for varying amounts of training data.

We observe that our proposed model achieves better performance than the baseline on the test set for all evaluated data set sizes. While the baseline performs similarly to ETNN on the training data for data sets of size 1,000 and 10,000, it clearly overfits and does not work well on the

test data—this emphasizes that our proposed model benefits from directly including the desired invariances into the models' architectures. As the baseline model has to learn about the invariances from the training data, its performance is particularly low for small training sets and it only reaches a maximum coefficient of determination of about -0.5 for data set sizes up to $1{,}000$. Our proposed ETNN already achieves a positive coefficient of determination for data sets of size $1{,}000$ and a coefficient of determination of 0.75 for data sets of size $10{,}000$.

5. Conclusion

We proposed and analyzed permutation trees, a hierarchical representation of permutations from which invariant maps can be derived. To this end, we introduced a sufficient condition on the structure of permutation trees guaranteeing the group property of the represented permutations. We show how to learn functions invariant to such permutation groups from the permutation tree in a bottom-up fashion using existing neural architectures for its internal nodes. Experiments on a synthetic Ferris wheel dataset show the effectiveness of our approach.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Amir, T., Gortler, S., Avni, I., Ravina, R., and Dym, N. Neural injective functions for multisets, measures and graphs via a finite witness theorem. *Advances in Neural Information Processing Systems*, 36, 2024.

Booth, K. S. and Lueker, G. S. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, December 1976. ISSN 0022-0000. doi: 10.1016/s0022-0000(76)80045-1.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Fuchs, F. B. and Veličković, P. Universality of neural networks on sets vs. graphs. In *ICLR Blogposts 2023*, 2023. URL https://iclr-blogposts.github.io/2023/blog/2023/sets-and-graphs/. https://iclr-blogposts.github.io/2023/blog/2023/sets-and-graphs/.

Gaiński, P., Koziarski, M., Tabor, J., and Śmieja, M. *Chi-ENN: Embracing Molecular Chirality with Graph Neural Networks*, pp. 36–52. Springer Nature Switzerland, 2023. ISBN 9783031434181. doi: 10.1007/978-3-031-43418-1_3.

Maron, H., Litany, O., Chechik, G., and Fetaya, E. On learning sets of symmetric elements. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6734–6744. PMLR, 2020. URL http://proceedings.mlr.press/v119/maron20a.html.

Tharmalingam, L. Sleep Health and Lifestyle Dataset. https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset, 2023. Accessed: 2024-06-03.

Wagstaff, E., Fuchs, F. B., Engelcke, M., Osborne, M. A., and Posner, I. Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151):1–56, 2022. URL http://jmlr.org/papers/v23/21-0730.html.

Wang, R., Albooyeh, M., and Ravanbakhsh, S. Equivariant maps for hierarchical structures. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pp. 13806–13817, 2020.

Wei-Kuan, S. and Wen-Lian, H. A new planarity test. *Theoretical Computer Science*, 223(1–2):179–191, July 1999. ISSN 0304-3975. doi: 10.1016/s0304-3975(98)00120-0.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, October 2018. doi: 10.48550/ARXIV.1810.00826.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets. *Advances in neural information processing systems*, 30, March 2017. doi: 10.48550/ARXIV.1703.06114.

A. Data Set Creation

The considered Ferris wheel data set is constructed by creating n samples as follows: (1) sample $n_G \cdot n_P$ passengers with characteristics according to the *Sleep Health and Lifestyle Dataset* data set (Tharmalingam, 2023), where n_G is the number of gondolas and n_P the number of passengers in each gondola; (2) randomly assign these passengers to gondolas; (3) compute a label (score) as described below and associate it to the sample.

A.1. Label Computation

For each sample (assignment of passengers to the gondolas), we compute a score (SCORE) as an artificial measure for quantifying the mood/satisfaction of the passengers and use it as the sample's label. The score generation is such that it satisfies the invariances according to the permutation tree in Figure 1b but is arbitrary otherwise.

The score is computed from individual gondola scores (GONDOLASCORE_i, where i is the index of the gondola) including a bonus for each gondola (BONUS_i) if neighboring gondolas have a similar passenger age group (intuitively accounting for the meaning that a larger group was not separated by other groups in between):

$$\text{SCORE} = \sum_{i=1}^{N_G} \min(10, T_i),$$

where

$$T_i = \left(\text{GONDOLASCORE}_i + \frac{\text{GONDOLASCORE}_{(i-1) \bmod N_G} + \text{GONDOLASCORE}_{(i+1) \bmod N_G}}{2}\right)/2 + 2 \cdot \text{BONUS}_i,$$

$$\text{BONUS}_i = \begin{cases} 1 & \text{if } \left|\text{AVGAGE}_{(i-1) \bmod N_G} - \text{AVGAGE}_i\right| < 5, \\ 0 & \text{else,} \end{cases}$$

where AVGAGE_i is the average age of the passengers in the ith gondola.

Note that SCORE satisfies the properties of a C-node. The computation of the gondola score (GONDOLASCORE_i) is described below.

Gondola score. For each gondola i we compute the gondola score GONDOLASCORE_i. For brevity, we omit the subscript i in the following. Let $p_1, \ldots, p_P$ be the passengers in gondola i. The gondola score is computed as

$$\text{GONDOLASCORE} = 2.5 \cdot \text{AGESCORE} + 2 \cdot \text{GENDERSCORE} + 0.5 \cdot \text{SLEEPSCORE} + 2 \cdot \text{BMISCORE} + 3 \cdot \text{FEARSCORE},$$

where the AGESCORE, GENDERSCORE, SLEEPSCORE, BMISCORE and FEARSCORE are based on the passengers' age, gender, sleep, BMI, and fear, respectively, as described below.

- Age score AGESCORE. The age score is constructed such that the more similar the ages of the passengers in a gondola are, the higher the score:

  $$\text{AGESCORE} = \begin{cases} 1 & \text{if gap} \leq 10 \\ (-1/60) \cdot (\text{gap} - 10) + 1 & \text{else} \end{cases}$$

  where

  $$\text{gap} = \max\{\text{age}(p_j) \mid j = 1, \ldots, P\} - \min\{\text{age}(p_j) \mid j = 1, \ldots, P\}$$

  and $\text{age}(p_j)$ is the age of passenger p_j.

- Gender score GENDERSCORE. This score is based on the distribution between female and male passengers in a gondola. It increases if the distribution is equal and reduces if the distribution is skewed towards one group. Concretely,

  $$\text{GENDERSCORE} = \begin{cases} 2 \cdot \min(\text{perc}_m, \text{perc}_f) & \text{if perc}_m \neq 0 \wedge \text{perc}_f \neq 0 \\ 1 & \text{else,} \end{cases}$$

  where perc_m and perc_f are the percentages of male and female passengers in the gondola, respectively.

- Sleep score SLEEPSCORE. The sleep score quantifies how well-rested the passengers in a gondola are:

$$\text{SLEEPSCORE} = \min(1, \text{AVGSLEEPQUALITY}/10 \cdot \text{AVGSLEEPDURATION}/7.5),$$

  where AVGSLEEPQUALITY and AVGSLEEPDURATION are the average sleep quality and average sleep duration of the passengers of a gondola, respectively. This quantifies whether the passenger is too tired to have fun and they would benefit from more sleep.

- Fear score FEARSCORE. The fear score quantifies the fear of passengers in a gondola based on their health information. The score is 0 if the product of the two blood pressure scores and the heart rate (using the maximal value of all participants in the gondola) exceeds the value $900{,}000$ and 1 in case it does not. Concretely,

$$\text{FEARSCORE} = \begin{cases} 0 & \text{if } \max_j(\text{blood-p1}(p_j) \cdot \text{blood-p2}(p_j) \cdot \text{heart-rate}(p_j)) \geq 900000 \\ 1 & \text{else,} \end{cases}$$

  where blood-p1(p_j), blood-p2(p_j) and heart-rate(p_j) are the two blood pressure scores for passenger j and the heart rate of passenger j, respectively.

- BMI score BMISCORE. The BMI score quantifies the possible discomfort of passengers in a gondola considering limited space and seating as follows:

$$\text{BMISCORE} = 1 - \frac{1}{N_P} \cdot \sum_{j=1}^{N_P} \max(0, (\text{BMI}(p_j) - 1)/2),$$

  were BMI(p_j) is the jth passengers BMI.