Hierarchical properties in merge trees for symmetry detection

Hierarkiska egenskaper i merge-träd för identifiering av symmetrier

Amanda Bigélius

Supervisor: Signe Sidwall Thygesen
Examiner: Ingrid Hotz

Linköping University | Department of Media and Information Technology
Master's thesis, 30 ECTS | Medieteknik
2025 | LiU-ITN-TEK-A--24/048--SE

Contents

Abstract

This work defines topological properties and aspects of merge trees which are relevant for detection of topological symmetry in merge tree representations of crystalline materials. In order to define these properties, three distinct perspectives on symmetry axes are presented to differentiate symmetry for different contexts. Namely, explicit axis, semi-implicit axis and implicit axis perspective. It is motivated why symmetry detection approaches based on the explicit axis perspective are not applicable to the context of topological structures in crystals due to unpredictable rearranging of initially symmetrical hierarchies. Instead, an alternative method based on implicit axis perspective is suggested for detecting symmetries in merge tree representations of crystalline materials. It is also discussed why merge tree representations only enable partial detection of specific geometrical structures in corresponding isosurfaces, such as layers and chains of atoms in crystals.

1 Introduction

1.1 Background

Layered 2D materials are used in many products such as solar cells and conductors. But the process of physically creating and testing materials individually is expensive and labor intensive. To efficiently find new material candidates for applications, a group of material scientists at Linköping University and the Technical University of Denmark consisting of Davidsson et al. [DBTA23] simulated a dataset of layered 2D materials. The material database contains several thousand materials which consist of different layered crystal materials.

To analyze the simulated materials, another project at Linköping University was carried out by Thygesen et al. [TAS+23] to develop a compact visual representation of the materials to enable intuitive visual analysis and comparison of the materials and their properties. The resulting representation was a collapsible merge tree and a corresponding bar representation. A merge tree is a topology-based representation that captures how separated isosurfaces in a scalar field emerges and merge together with regards to changes in the scalar value. In this context, the scalar values represents charge density within the material and the merge tree was used to visualize topological changes within the crystal structures. Namely the formation of bonds between atoms or groups of atoms.

Due to the crystalline properties in the simulated materials, the merge tree representations of the materials are highly symmetrical. When additional atoms are added to create new variants of the material, this impurity shows up in the merge tree as asymmetric branches.

To efficiently distinguish characteristics of the materials it is of importance to identify the impurities, its properties and its general effect on the material. A visual approach to do this is to identify symmetries in the merge tree and collapse them to isolate the asymmetric outlier. Combined with the said bar representation, this approach would result in a compact visual representation and enable overseeable side by side comparisons of impurities in different materials.

It is also of interest to know which saddle nodes bind together larger geometrical structures in level sets, such as layers, since the corresponding scalar value is related to the conductivity in the material. But due to the size and complexity of the trees and the database, manual identification of symmetries and specific nodes is practically unfeasible.

1.2 Project aim

The aim of this project is to develop a robust method to automatically identify symmetries in these merge tree representations. Robustness within this context refers to a stable and consistent method which reduces the negative impact of noise, structural changes and large value differences within the data. To identify symmetries it is also necessary to define symmetry within the given context of merge trees and material science. The aim also includes evaluating possible relations between structural properties in the merge tree and the formation of larger geometrical structures in isosurfaces, such as chains and layers, for the layered materials.

The purpose of this work is to enable interactive and collapsible visual tree representations with regards to symmetry properties. This would make the representation more compact, as well as enable fast identification of symmetries and outliers, which further supports the sifting of promising material candidates. While the main focus of this project is to develop theories and methods for defining symmetry in the given context, the end goal of the research is to enable the implementation of symmetry detection for merge trees as a feature in the software Inviwo.

1.3 Research questions

1. Which structural properties of the merge tree representation defines symmetry in charge density fields?
2. Are branch decomposition-based symmetry detection approaches applicable within the domain context of molecular analysis of charge density?
3. Is it possible to automatically detect and distinguish larger geomterical crystal structures such as chains and layers topologically from the merge tree representation?

1.4 Research approach

The definition of symmetry is the existence of similarity around an axis. Symmetric similarity in the context of merge trees can be further divided into two different categories of symmetry factors: Structural symmetry factors in the tree representation, such as similarities in the positions of nodes and branches, and attribute symmetry factors within the data which consist of similarities in attribute values specific for individual nodes. An example of such an attribute in the given context is the atom type.

However, there is no universal definition of what symmetry is for an arbitrary multivariate dataset, or how a symmetry axis is defined for that matter. Depending on perspective and abstraction level, any common parameter or branch in the merge tree could technically be considered as a symmetry axis. Furthermore, all attributes in the data might not be of symmetric relevance. So the design of a symmetry model is highly dependent on the specific data and the purpose of the symmetry model.

The first step of the investigation was therefore to define the relevant symmetry factors for the given application. Based on these factors an artificial merge tree was created as a simple base case. This tree was designed to be fully symmetrical with regards to the relevant symmetry factors so that it was possible to easily test whether or not an applied symmetry detection method resulted in the expected levels of symmetry. An already existing method for symmetry detection in scalar fields, proposed by Natarajan et. al. [TN11] was applied to the base case model to see whether it was applicable or not with regards to the previously defined symmetry factors. The results of the application was then analyzed to identify and define potential missing factors in the proposed symmetry model definition, as well as to find potential shortcomings with the existing symmetry model and detection method with regards to the given context. The results were then used as a basis for designing a more suitable method for detecting symmetry in merge tree representations of crystal materials.

1.5 Project scope

Due to the highly context-dependent nature of the model, only tree symmetry in the context of topological material evaluation is considered. Consideration for other domain specific uses of merge trees and symmetry in merge trees will not be taken into account unless it is of relevance for the given context.

The main point of interest is structural symmetry explicitly derived from the merge tree. Properties of external variables with possible relevance for symmetry, but which are not included in the merge tree representation itself, will be left out of the current method for symmetry detection. Although, some will be mentioned for the sake of theoretical context and discussion.

Due to the given time limitations and for the sake of structured and controlled evaluation, only small synthetic materials will be used for testing. Larger and more complex real-life data from the database will not be used for testing unless it is ensured that a method works on smaller controlled cases first.

The task of optimizing any implementation of symmetry detection will be left entirely to future work. The main task is to define symmetry within the given context and to design and evaluate a method for extracting it from the merge tree. The task of implementing the interactive visualization is also left for future work.

2 Theoretical background

2.1 Level sets

A level set $L_\alpha(f)$, also called a "contour", of a function f is the set of values x for which $f(x) = \alpha$. For a function $f(x, y, z)$ representing a 3-dimensional scalar field, the level set $L_\alpha(f)$ contains all space coordinates (x, y, z) where $f(x, y, z) = \alpha$. Interpolation over adjacent values in the level set creates an isosurface which outlines the shape of the level set in space. A level set can contain several unconnected isosurfaces which can be either open or closed. In this project, all isosurfaces in a level set are closed and can be interpreted as the outlines of atoms and bonds between atoms as seen later in figure 6.

2.2 Merge tree representation

The merge tree representation [YMS+21] is a topological data representation of a scalar field. It represents how individual isosurfaces in a level set split or merge with each other when the scalar value α in the field changes, as seen in figure 1. While a merge tree can represent topological behavior in any dimension it will only be used to represent topological changes in 3-dimensional isosurfaces during this project. In the given context, α is the charge density within the material.

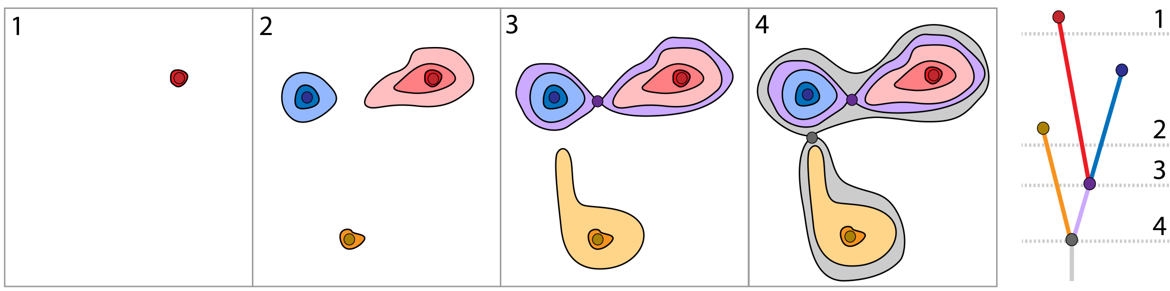

Figure 1: Visualization of how the merge tree corresponds to level sets in 2 dimensions. Used with courtesy of Signe Sidwall Thygesen.

Leaf nodes in a merge tree signify the moment an isosurface either begin or ceases to exist in the α parameter space, which is seen on the appearing contours in visualization (1) and (2) of the figure. Saddles between two edges represent the merging or splitting of different isosurfaces as seen in visualization (3) and (4). The lifespan of an isosurface, or the length between its leaf and root in the merge tree, is denoted as its persistence.

A child node is formally denoted as a "down node" in relation to its adjacent parent node. A parent is similarly denoted as an "up node" in relation to its child node. Despite this terminology, merge trees will be depicted with the root node at the bottom throughout this project. A sub-tree in this context denotes a subset of edges in the merge tree which contains a local highest parent node and all descendants below it.

2.3 Branch decomposition

Branch decomposition is a state of the art approach of processing a merge tree to enable further analysis with regards to the persistence of paths as described by Saika *et al.* [SSW14].

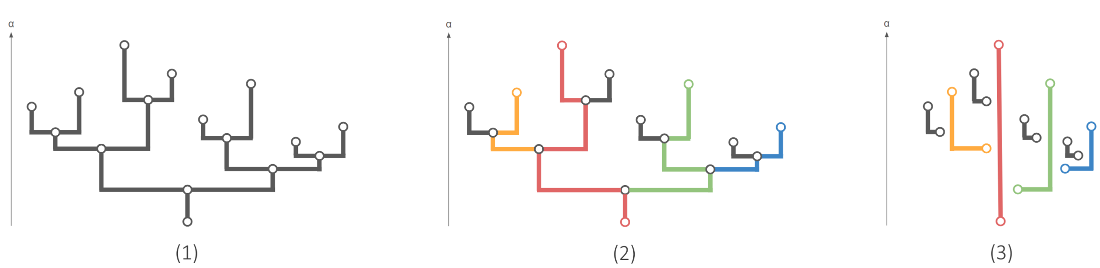

Figure 2: Infographic illustrating how a merge tree is split into a branch decomposition. Graph 1 is the initial tree. Graph 2 is a visualization of the longest paths in tree and graph 3 is the corresponding branch decomposition.

It decomposes the entire tree into individual branches with regards to the longest path from root to leaf, as seen in figure 2. The path with the longest possible persistence becomes the main branch, while branches with shorter persistence either becomes children to the main branch, such as the green and yellow branch in the figure, or end up further out in the hierarchy like the gray branches. The resulting branch decomposition tree is a more compact representation of the data than the initial tree structure since it contains fewer edges and it can be ordered with regards to the persistence of its branches. Each decomposed branch can be summarized as the path between a unique leaf node and a saddle node where the branch attaches to a longer branch, or to the root of the tree.

2.4 Branch decomposition-based similarity measurement

Thomas and Natarajan [TN11] previously proposed a symmetry model, as well as a method to detect such symmetries, in scalar fields based on decomposed merge trees. It consists of cross-comparing the separated branches in the decomposed merge tree and then assigning them to symmetry groups according to similarity.

To make the comparisons compact, each decomposed branch b is represented by a hierarchy descriptor $h = (D, U, <G_1, \ldots, G_i>)$ where D is the leaf node scalar value ("down node") of the branch, U is the root scalar value ("up node") of the branch and the vector $<G_1, \ldots, G_i>$ is the list of adjacent child branches, defined by their assigned symmetry group. The descriptor is visualized in figure 15 and 16 which contains visualizations of different branches and their hierarchy descriptors h. This representation contains the connectivity of all nodes and unique sub-trees in the merge tree in a bottom-up manner. It also keeps the size of the representation relatively small since each sub-trees is defined in terms of the symmetry groups it contain, instead of explicitly storing all child branches or edges anew for each unique subset.

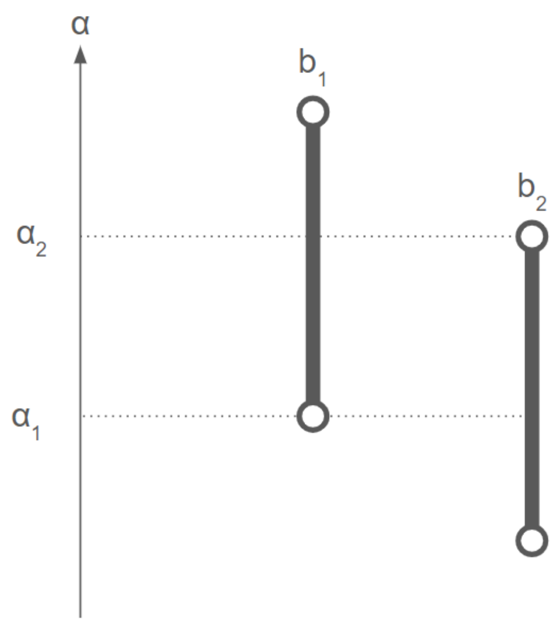

Figure 3: Illustration of overlap between two branches b_1 and b_2.

Symmetry between two sub-trees is defined by the scalar overlap O of the main branches as well as all their children. As seen in figure 3, the overlap between two branches $b1$ and $b2$ is defined by the distance $\alpha_2 - \alpha_1$, where α_1 is the highest root and α_2 is the lowest leaf in the pair of branches. To get a percentage of how much overlap there is with regard to total persistence, the distance $\alpha_2 - \alpha_1$ is doubled since they represent overlapping intervals in two different branches. The total overlap value is therefore defined as $O = 2 \cdot (\alpha_2 - \alpha_1)$.

If two individual branches have 100% scalar overlap then O is equal to the sum of persistence (total life duration) of b_1 and b_2 and the branches are assumed to be 100% symmetrical. The symmetry metric S for sub-trees containing several branches can be read in equation 1.

$$S = \frac{O_1 + O_2}{p_s + p_t} \cdot 100 \tag{1}$$

The similarity metric S is the weighted percentage of overlap between the sub-trees s and t with regards to their combined persistence. O_1 is the overlap between the main parent branches representing the sub-trees and O_2 is the maximum assignment solution of overlap between each sub-tree's set of child branches. The overlap O_2 is calculated by creating pairs of child branches, one from each sub-tree, from which partial overlaps are calculated. Each child branch can at most be part of one pair (sometimes none if the sets have different sizes), but depending on which child branches we pair up and calculate overlap between, the final overlap value O_2 can vary widely. To enable an exact and stable similarity score, O_2 is defined as the largest possible sum of overlaps between the two sets of child branches. To find this optimal overlap, a maximum assignment method is used to find the optimal unique pairs of child branches. The variables p_s and p_t are the total sum of persistence in each sub-tree s and t. This approach ensures that two branches with highly similar structures, persistence and scalar intervals will get a similarity score S close to 100.

Symmetry groups are defined by the hierarchy descriptor of the first branch assigned to them. If a branch has >90% similarity with an existing group descriptor, it will be assigned to that group. But if a given branch does not match an existing symmetry group to the required extent, a new group will be created based on this branch's hierarchy descriptor instead. The algorithm iteratively compares each branch in the tree to all existing symmetry groups until all branches are assigned to a group. When all symmetry groups are created, the group assignments are refined in a post-processing step where each branch is compared to all other existing symmetry groups. If there exists a group with a higher similarity score than the one which is currently assigned, the branch is assigned to that group instead.

To avoid repeated and recursive comparison of child branches, the overlap between the symmetry groups from the initial comparison is stored in a similarity matrix to enable easy access for future comparisons.

Branches are processed shortest — or utmost — first, so the shortest branch will by default become the blueprint for the first group. This ensures that all child branches in a sub-tree are assigned to a group before the parent — which depends on its children's descriptors — is processed. Initial checks are made to set similarity to 0 without cross-comparison if the up node of one branch is greater than the down node of the other and vice versa, since that means that there is no overlap between the branches.

2.5 The Munkres algorithm

Munkres algorithm [Mun57], or *the Hungarian algorithm*, is a method to solve the minimum or maximum cost assignment in a complete, weighted bipartite graph. In this project it is used to solve the optimal pairing of child branches for O_2 in the similarity measurement in section 2.4.

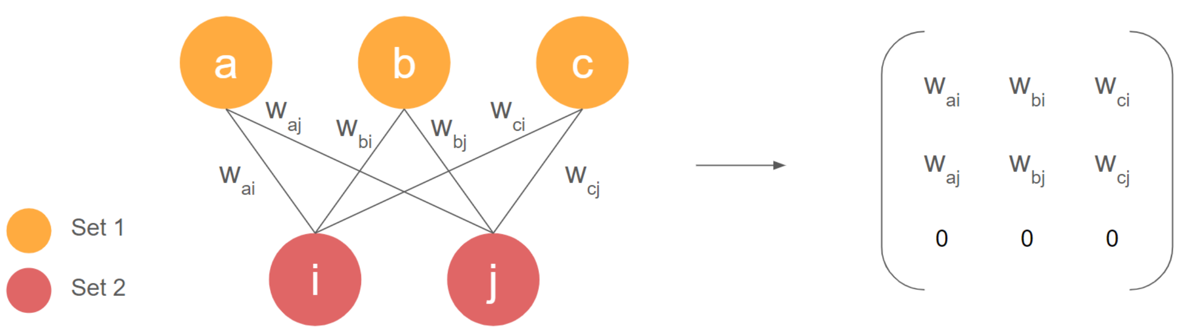

Figure 4: Complete bipartite graph formed by two datasets (left) and its matrix representation (right)

A complete bipartite graph is a data structure where each element in a set connects to every other element in another set as seen in figure 4. In the given context the graph represents the two different sets of child branches that are compared to calculate O_2. The weighted connections w between them are the overlap value for each possible unique pair.

In practice, the assignment is is done by creating a quadratic matrix of the dimension n^2 out of all the possible overlap values w and then iteratively re-balancing it to find a possible solution to the maximum overlap problem which contains optimal pairs. The resulting value from the algorithm is the largest possible sum of overlap values from the optimal unique pairs. This includes the restrictions that a pair consist of branches from different sub-trees and that each branch only can be part of at most one pair in the final result as described in section 2.4.

The matrix re-balancing for a minimum cost assignment consist of the following steps:

1. Subtract the local minimum value from each row to create zeros, which correspond to possible optimal solutions.
2. Subtract the local minimum value from in each column to create zeros.
3. Cover all zeros in the matrix with the least possible amount of horizontal or vertical lines and check if the number of lines is equal to n.
4. If the number of lines is less than n, then the pairs are not evenly distributed and the current solution does not include all unique branches. Find a new min value among all the uncovered elements and subtract it from the uncovered elements.
5. Add the same minimum value to any covered element located in a line crossing.
6. Repeat step 3 — 7 until the number of lines is equal to n to ensure that all branches are included.
7. Mark a zero from the column or row with the least zeros as optimal and remove any other zero in the same row or column as the marked zero to ensure that each branch only occur in one pair.
8. Repeat 9 — 10 until there is exactly one marked zero in each row and column. Each zero then represent the optimal solution located in the corresponding pair.
9. Use the index of the final marked zeros to obtain the corresponding optimal weights from initial matrix of weights

If the sets have different sizes, the matrix is padded with zeros to represent the missing values. To solve for maximum assignment instead for O_2, the problem is converted into a minimum assignment problem by replacing the weights $w_{i,j}$ in the matrix with the value $(w_{MAX} - w_{i,j})$, where w_{MAX} is the global maximum weight in the matrix. If the dimensions are non-matching, the matrix is padded before the conversion which results in a padding of max values instead of zeros.

2.6 Pre-processing methods

One method to make the measurement more robust is to pre-process the tree by removing branches with a persistence lower than a given threshold to filter out potential noise. The threshold applied by Thomas and Natarajan [TN11] is 1% of the longest branch's persistence.

The hierarchy of a branch decomposition is also highly sensitive to the placement of the saddles and slight changes in their placements, due to noise or sensor errors, can alter the hierarchy and the resulting symmetry score.

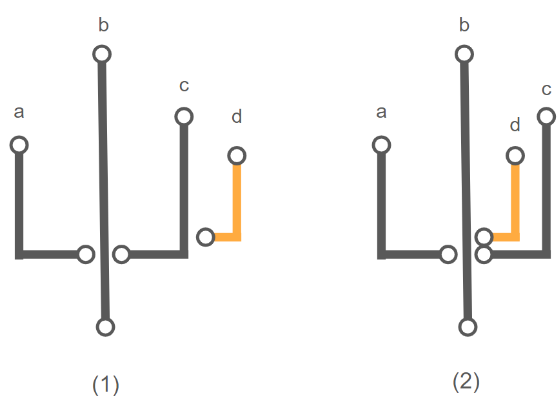

Figure 5: Left: Branch decomposed merge tree where branches c and d are considered unstable. Right: Same merge tree after applying saddle stabilization.

A possible solution to avoid unnecessarily deep and sparse hierarchies, and lessen the impact of the structural differences on the symmetry detection, is to apply saddle stabilization. This method traverses the tree from leaf to root and rearranges saddle points that are very close in height so that previous children to a branch becomes siblings instead. By doing this the structure is kept compact and close to the main branch. As seen in figure 5, saddle stabilization is applied by turning an unstable outer branch d in (1) into a sibling of its previous parent in (2) by moving it up one step in the hierarchy. This notably changes the similarity score between branch a and c since the symmetric influence of branch d is removed which makes them much more likely to be classified as similar.

3 The data

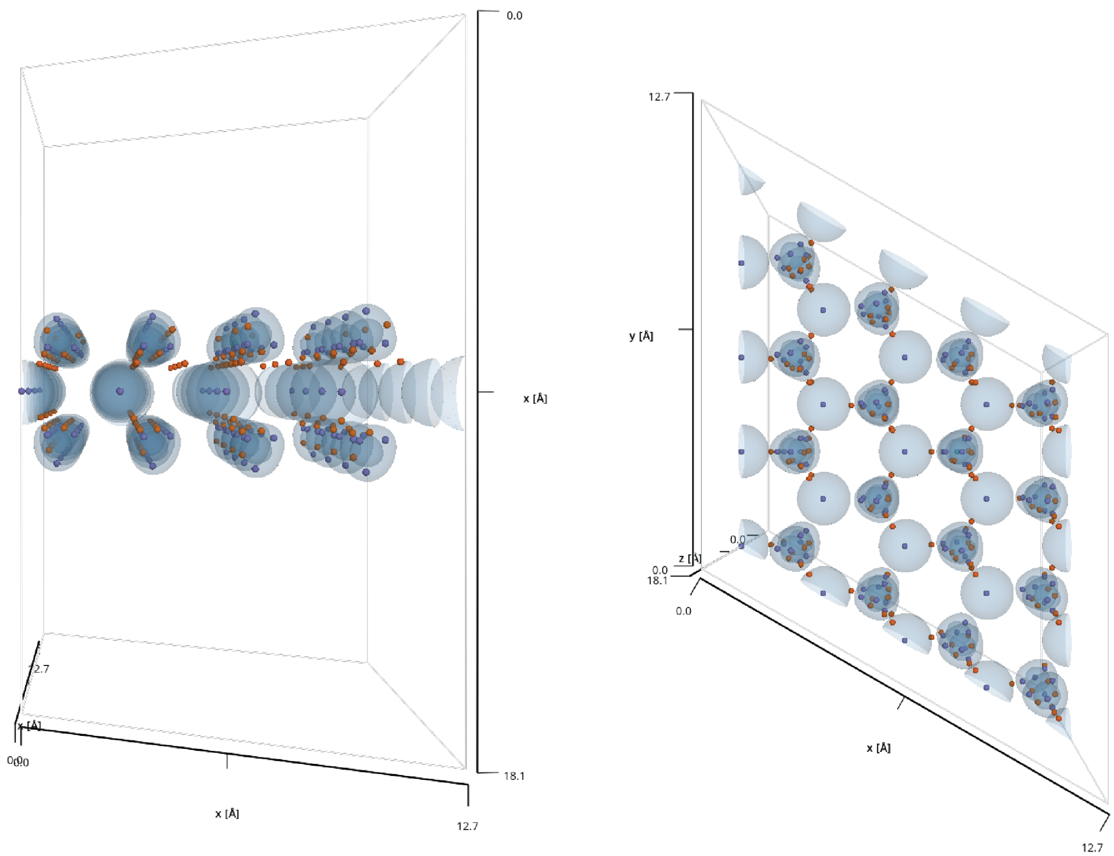

Figure 6: Level set visualization of isosurfaces and critical points in a unit cell of the crystal MoS2. The blue points are local maximum values located within atoms and the orange points are saddle points which connects different isosurfaces. The critical points are visualized independently from the α value, hence the free-floating saddle points in the left image

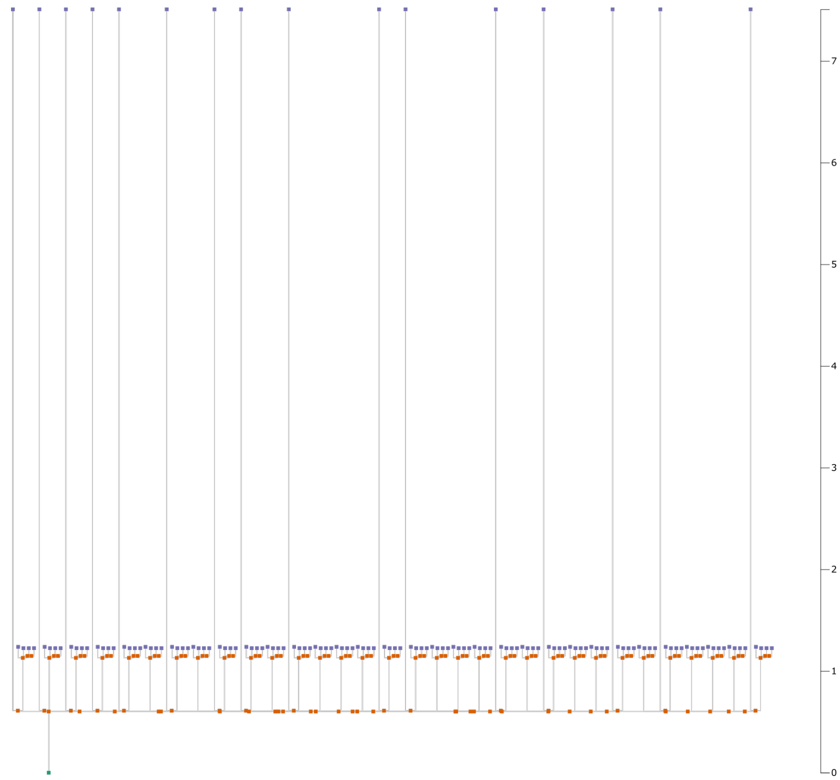

Figure 7: Merge tree representation of the crystal MoS2 with local maxima (blue) and saddles points (orange)

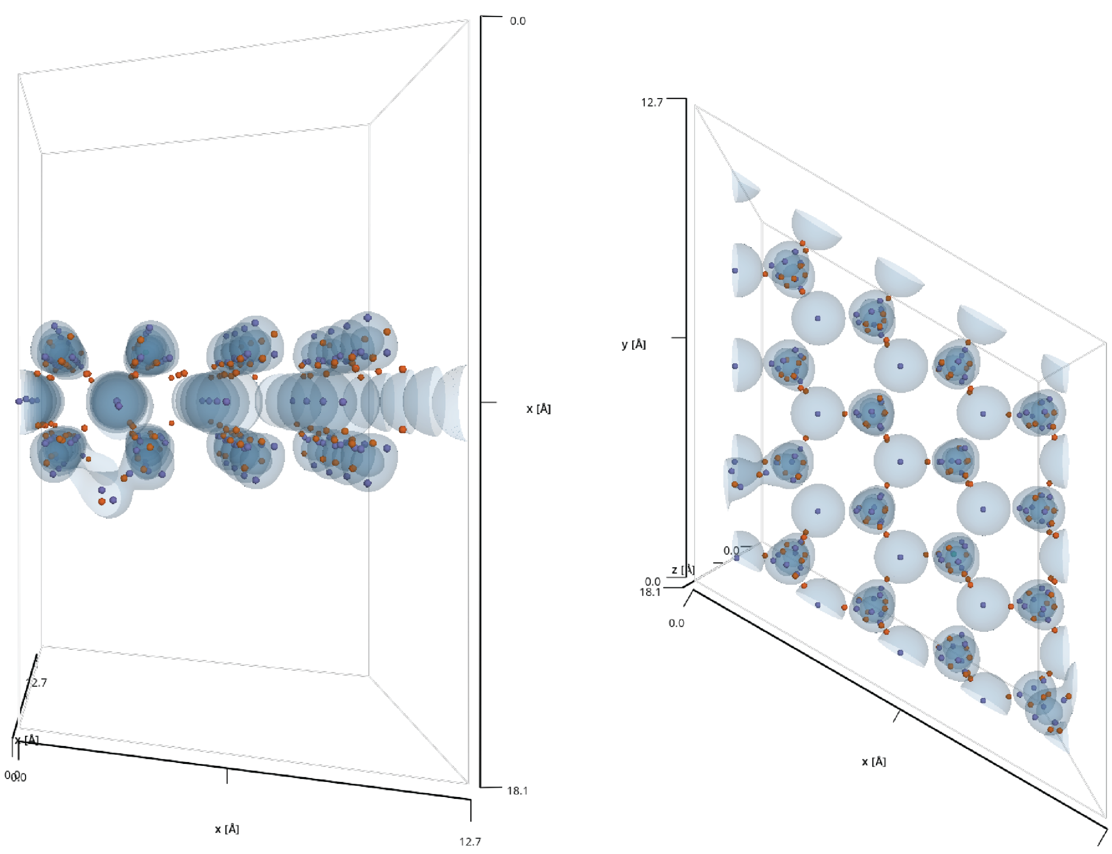

Figure 8: Isosurface visualization of a unit cell subset which contain the crystal structure of MoS2 with an additional N atom. Local maxima (blue points) and saddles (orange points) are visualized independently from the α value. Hence the occurrence of free-floating saddle points

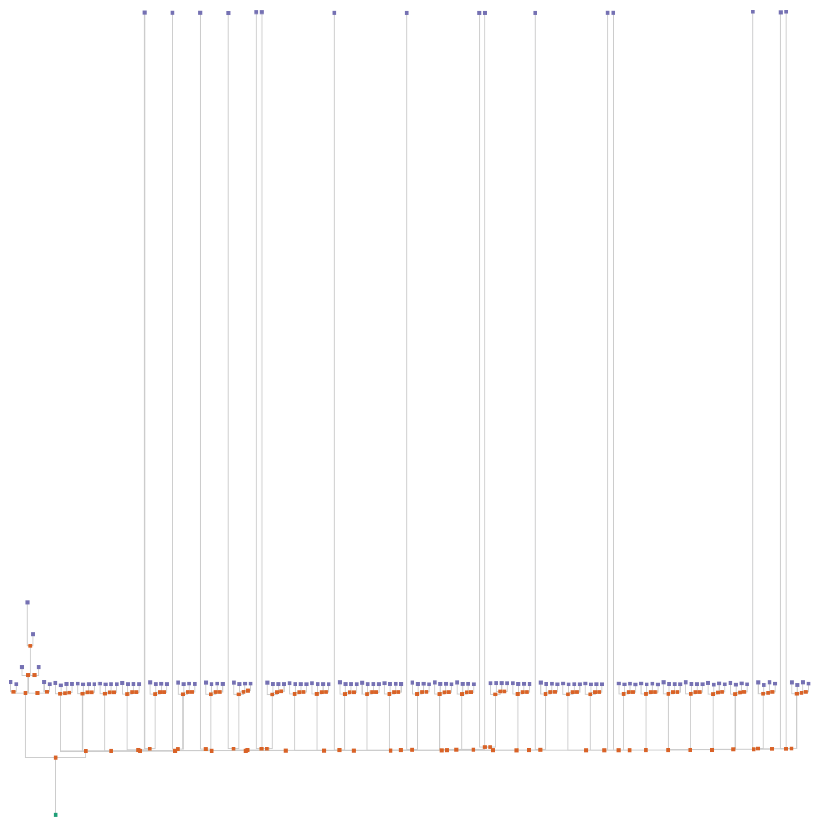

Figure 9: Merge tree representation of the crystal MoS2 with an additional N atom

The material database produced by Davidsson et al. contains experimentally known crystalline 2D monolayer materials and polluted, or *doped*, variants of these materials where additional atoms, or *impurities*, had been added to some part of the crystal structure.

The materials are defined by their atom positions and charge density. The charge density is represented as a 3-dimensional scalar field which contains the sampled distribution of electrons on a grid. Local maximum values in the field corresponds to the positions of individual atoms or subatomic objects. The maxima values, as well as the saddle points, can be seen as blue (maxima) and orange (saddle) dots in the isosurface representation of MoS2 in figure 6. The surrounding isosurfaces can be interpreted as the spatial outline of atoms and the bonds between them. The density is highest at the center of the atoms and successively gets lower further away from atom cores and in areas where the atomic bond strength decrease.

Due to a steep exponential increase of charge density close to the center of atoms, the calculations of the charge density are simplified in a way such that the resulted simulated maxima is notably lower than in the actual atomic nuclei. As a result of this simplification it is also possible for a simulated atom to contain several maxima. The possibility of a maxima not necessarily representing the actual value or position of an atomic nucleus should therefore be taken into consideration when interpreting these values. The merge tree for the material MoS2 can be seen in figure 7 where the merging pattern of the branches have a clear visual correlation to the bonding patterns in the corresponding isosurfaces in figure 6.

Figure 8 is an isosurface representation of MoS2 with an additional nitrogen atom. Identifying the impurity and its characteristics directly by viewing the iso surface is difficult. But when the data is visualized as a corresponding merge tree as seen in figure 9, the impurity clearly shows up as an asymmetric outlier to the left.

The database is categorically sorted in terms of material name, site of the impurity and type of impurity. It stores further physically interesting key-value pairs related to the materials such as formation energy, depth parameter, layer expansion factor, spin and magnetism etc. But this specific project is limited to the charge density data.

The pattern in crystalline structures repeats itself periodically in all directions which makes it redundant to store and evaluate the entire material structure. To avoid this, a unit cell representation of the material data is stored instead. A unit cell representation is a spatial and periodically repeating subset of the material structure with the size of approximately 10Å. It is specified to capture all unique characteristics in the material and at least one impurity [DBTA23]. This unit cell can then be repeated — or *expanded* — periodically in space to create arbitrary large material models on demand. The expanded representation which contains several repetitions of the unit cell pattern is called a supercell. By estimation from Davidsson, the largest unit cells in the dataset contain around 60 different charge density maxima, which corresponds to leaf nodes in the merge tree representation. Since all unique patterns and possible symmetries in the material structure are contained within the unit cell, the symmetry detection method is only applied to unit cells in order to minimize calculations.

Charge densities close to an impurity are affected due to the influence of the impurity bond. This influence is prominent in the unit cell of simulated materials since it represents a very small subset of the material. But in real materials there might be larger distances between the impurities and the influence will be considerably smaller far away from the dopant sites. This is important to consider in terms of bias since the unit cell subset potentially represents a volume with disproportionate scalar displacement compared to the rest of the material.

Node ID	Value	Critical type	Parent ID
0	136.621	3	59
1	136.621	3	56
2	136.621	3	54
3	136.621	3	52
4	138.631	3	38
5	138.631	3	39
...	...	...	...
63	1.47416	0	null

Table 1: Table showing the first 6 nodes and the root node in the data for the material BeN4. The critical types are leaf nodes (3), saddle nodes (2) and root nodes (0)

The data which is directly used for symmetry detection is a node table derived from the charge density level set. It contains child-parent relations, scalar values and the nodes critical type. The critical types in use are the global minima (the root), local maxima (leaves) and saddle points. These are represented numerically by integers in the table. The pipeline to produce this table was provided from the previous project by S. Thygesen et al. [TAS+23] revolving the merge tree.

4 Defining initial conditions for symmetric similarity in tree representations of material data

As mentioned in the section 1.4, the definition of symmetry is highly dependent on the given context and data. Within the context of identifying crystal formation patterns with regards to charge density, some factors and conditions can be established directly. One of the main material properties of interest with regards to symmetry is the bond strength between atoms and structures. This property corresponds to the charge density value α in saddle points and general similarities in scalar value. Subsets of isosurfaces — or branches in the merge tree — with little or no overlap in α are therefore not considered symmetrical. An example of this can be seen in figure 10, where sub-trees A and B should have no symmetry with regards to overlap. Therefore the α axis is the main symmetry axis within this context and the scalar overlap around it is the first condition for symmetry. Further conditions for symmetric similarity are the amount of nodes in each sub-tree and which atoms the nodes correspond to. For example, a sub-tree with 100 nodes should have very little similarity to a sub-tree with only 2 nodes. In figure 10, this means that sub-tree B and C should have relatively low symmetry. Similarly, sub-trees with no matching atoms should have little to no similarity, depending on the importance it has for the material properties. Therefore, sub-tree B and D should have no symmetry as well. To summarize these factors, the initial symmetry criteria for two sub-trees consists of

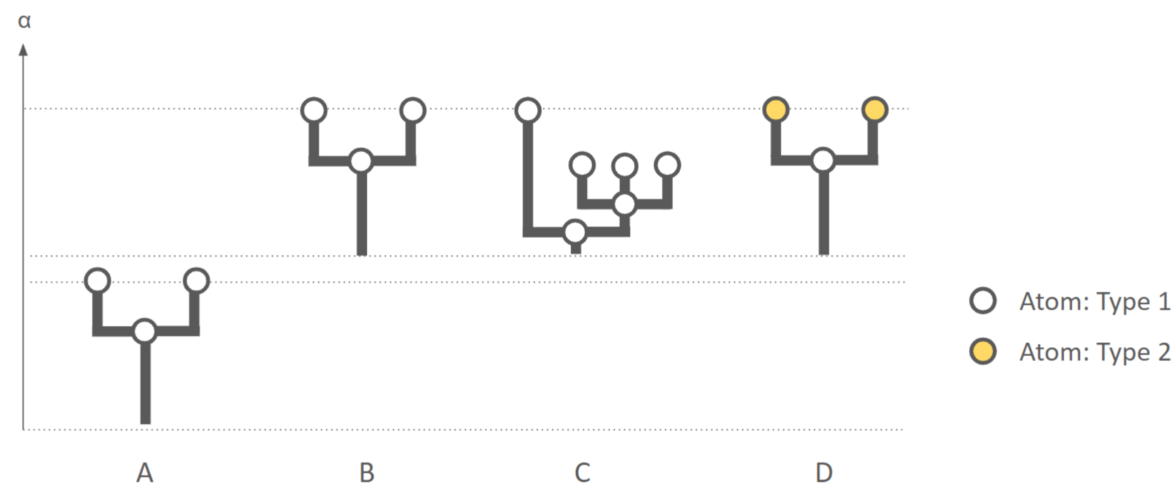

Figure 10: Illustration of the different factors representing symmetry in the given context (different example trees and whether or not they are symmetrical)

- Individual branch overlap with regards to the charge density value α
- Likeliness with regards to number of branches
- The maximum overlap between child branches
- Matching atom types or likeliness in other relevant node attributes

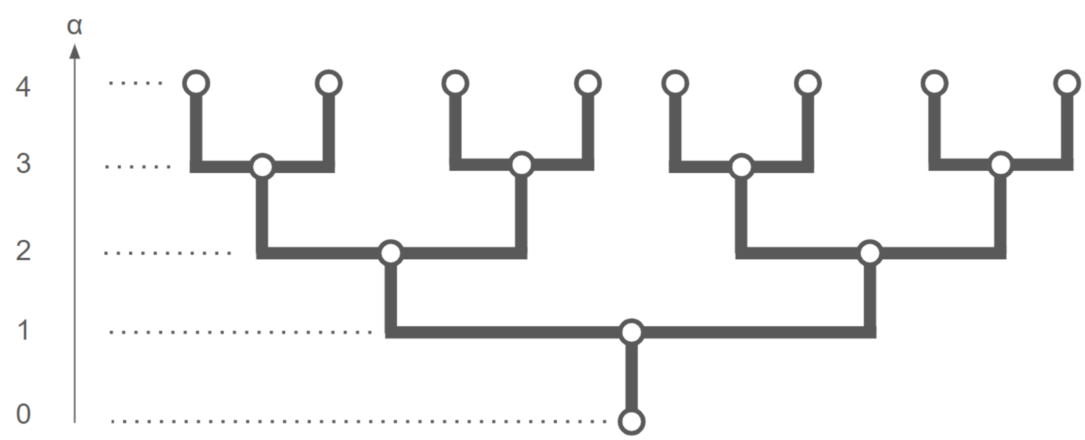

Figure 11: The artificial and completely symmetrical merge tree on which the symmetry detection method was tested

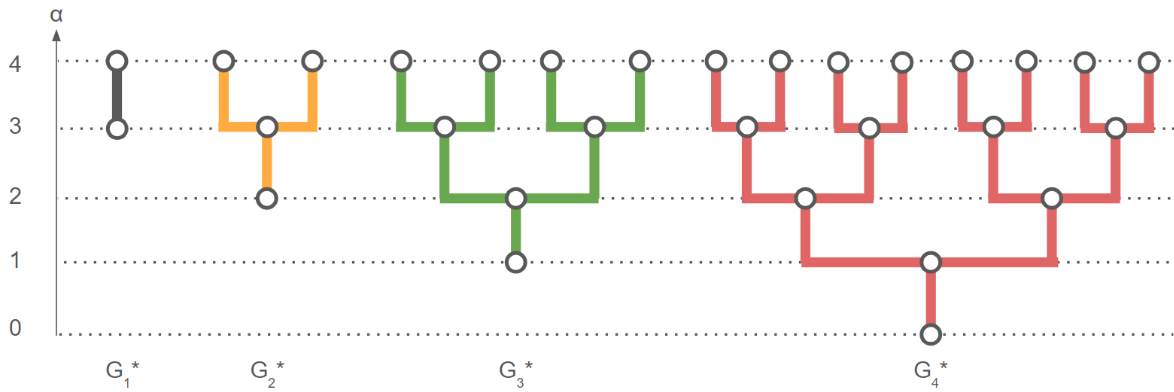

Figure 12: The expected symmetry groups G_i^* that should be obtained when applying symmetry detection on the artificial test case merge tree

Group	Occurrences
G_1^*	8
G_2^*	4
G_3^*	2
G_4^*	1

Figure 13: The expected number of occurrences per expected symmetry group G_i^*

To enable testing of these criteria, an artificial symmetrical merge tree was created as seen in figure 11. All sub-trees within the same scalar interval were designed to be classified as 100% symmetrical according to the symmetry criteria above. Applying a working symmetry detection method should result in four different symmetry groups as seen in figure 12. One for each layer of nodes, disregarding the root node. The expected occurrences for each group in the case tree is listed in table 13 for later comparison.

5 Implementation of existing symmetry measurement

This chapter explains the implementation of the method presented by Thomas and Natarajan in 2.4.

5.1 Software and initial workspace

The implementation was done in the Software Inviwo [SSK+15] which is a node-based scientific visualization tool developed mainly within the research group of Scientific Visualization at Linköping University. It works by connecting data nodes and different data processor nodes into a pipeline to process and visualize the data. The processors used for the project and the new one developed specifically for symmetry detection are written in C++ and depend on the external C++ library Topology Toolkit (TTK) [JGJGC18]. The software does not support the exporting or passing of nested data structures such as trees between processors. Instead, one of the primary ways of transferring data within and from the software is via dataframes, which is also the main method used for this project.

The implementation of a symmetry detection processor is a continuation of an existing processor pipeline which produced the merge tree node list in table 1, as described in section 3. This previous pipeline was provided via the merge tree project by Thygesen *et al.* [TAS+23]. It works by converting the initial material data format to volume data, adjusting it to the format of the volume library in use. The data is then filtered, processed and topologically simplified to acquire the node table in figure **??**.

5.2 Processor design

The new processor processes the existing node list and applies branch decomposition, as well as the existing symmetry detection schema to it. It then exports a modified version of the initial table in figure **??** as a dataframe with additional symmetry group identifiers G_i for each node. The purpose of these identifiers is to enable collapsing of entire subsets of branches in future visual representations of the merge tree.

Due to the lack of support for nested structure outputs, both branch decomposition and symmetry detection was implemented into the same processor to keep the tree structure intact, and the representation compact, between the two steps.

5.3 Implementation of branch decomposition

The non-recursive branch decomposition schema traverses the nodes in-order and iteratively connects leaf nodes upwards towards the root. All branches are initially treated as potential main branches until a deeper sibling is found. Then the shorter branch is ended and assigned as a child to the longer branch. To simplify the computations, branches were stored in a list as custom-made branch objects which contained the up and down nodes, persistence and parent of the decomposed branch. Since they indirectly contained the same information as the hierarchy descriptor (scalar value, parent, children), the branch objects were used in place of the original suggested hierarchy descriptor to avoid duplicate memory storage and unnecessary computations.

5.4 Implementation of symmetry detection

Branches (B)
b_1
b_2
b_3
...
b_i

Symmetry group descriptors (G)	
Id	Descriptor
G_1	b_a
G_2	b_b
G_3	b_c
...	...
G_i	b_j

Symmetry score matrix

1	$S_{(G1,G2)}$	$S_{(G1,G3)}$	...	$S_{(G1,Gi)}$
$S_{(G2,G1)}$	1	$S_{(G2,G3)}$	...	$S_{(G2,Gi)}$
$S_{(G3,G1)}$	$S_{(G3,G2)}$	1	...	$S_{(G3,Gi)}$
...	...	...	...	...
$S_{(Gi,G1)}$	$S_{(Gi,G2)}$	$S_{(Gi,G3)}$	...	1

Figure 14: The three containers used to implement symmetry detection. Left: List of decomposed branch objects b_i. Middle: List of symmetry group indexes G_i and the branches that define them. Right: Similarity matrix containing the symmetry scores S for all unique pairs of symmetry groups $< Gi, Gj >$

Three different data containers are required for the symmetry detection schema which are shown in figure 14; The list of branch objects B (sorted according to persistence), a list G to store the found symmetry group indexes G_i and descriptors b_j, and finally a quadratic similarity matrix to store the total overlap values S between all possible pairs of symmetry groups $< Gi, Gj >$. Initially, both the symmetry group list and similarity matrix is empty.

As previously mentioned in section 2.4, the descriptor of a symmetry group G_i is the first branch b_j which is assigned to it by using it as a representative for all other similar branches. This in turn makes the symmetry detection more compact and efficient since we avoid repeated definitions and comparisons for highly similar branches. For that reason, the list G consists of pointers to the corresponding representative branch objects b_j for each group. The execution work in the following order:

1. Sort the branch list B with regards to persistence.
2. Compare each branch, in ascending order of persistence, to each of the existing symmetry groups G_i by applying the similarity measurement function described in 2.4. The symmetry scores from these comparisons are stored in a temporary vector V for eventual later use.
3. If a group descriptor with >90% similarity is found, the branch is assigned the corresponding group ID and nothing is added to the descriptor list or similarity matrix. If several possible matches are found, the branch is assigned to the group with the highest similarity score.
4. If no compatible group is found a new similarity group is added to G and the current branch becomes its symmetry group descriptor b_j. This is done by appending V to the similarity matrix to expand it in both directions with the similarity scores of the new group, along with the self-similarity 1 on the diagonal. The first branch b_1 will by default become the the first symmetry group descriptor b_a.
5. When all branches in B have been assigned to a group, the assignments are refined in the post-processing step by finding groups with potentially higher similarity scores in the now complete similarity matrix. If a group with higher similarity to the branch is found, the branch's assigned group ID G_i is updated to that of the best compatible group.

After the final step, all branches are assigned a group ID G_i corresponding to a group with at least one branch in it. Similar to the similarity matrix, a matrix with to the total overlap values O or $(O_1 + O_2)$ between the groups were stored to aid with the implementation of the Munkres algorithm.

5.5 Implementation of the Munkres algorithm

The Munkres algorithm was implemented according to the schema in section 2.5. The overlap values which are used as weights are read from the overlap matrix described in section 5.4. But since this matrix contains the total overlap O or $(O_1 + O_2)$ for all possible pairs of branches — or rather, their group representatives — in the entire tree, and not just the children of the two branches of interest, this turns the subset of interest into a sparse matrix. To avoid nested iterations over a sparse matrix, the algorithm is applied to a temporary overlap matrix which only contain the scalar overlaps of the branches of interest. This matrix is no longer symmetrical and each dimension represent the list of children to one of the main branches. If these lists are of different sizes the matrix is padded with zeros as described in section 2.5. The reduction and balancing of rows and columns were done iteratively in a straight-forward manner and additional containers and trackers were implemented to keep track of crossed or marked values and lines. Optimality conditions were defined in terms of binary values and counters. An example of such a counter was the number of zeroes in each row and column.

The final iteration of the balanced matrix is expected to have exactly one zero — representing an optimal solution — in each row and column to ensure unique branch occurrences in the resulting solution. When a zero is marked as optional, any other zero in a crossing row or column was discarded to ensure this. The order in which the zeros are marked is given by the rows and columns with the lowest zero-counters. Choosing the optimal zero from the row or column with the lowest number of zeros ensures that the discarding of leftover zeros does not erase the only viable option in another row or column.

6 Results from applying the existing symmetry detection method on material data

This section includes the results of applying the previous symmetry detection method from 2.4 to the base case tree in figure 11. The groups derived from the initial base case merge tree is shown in figure 15. The method was also applied to two differently scaled versions of the base tree (figure 17, 19) to test whether or not the group descriptors and their number of occurrences were the same between the different cases. The resulting descriptors and occurrences for the three different cases are summarized in figure 16, 18 and 20. The detected symmetry groups in the scaled cases maintain the same structure and hierarchy as the unscaled version.

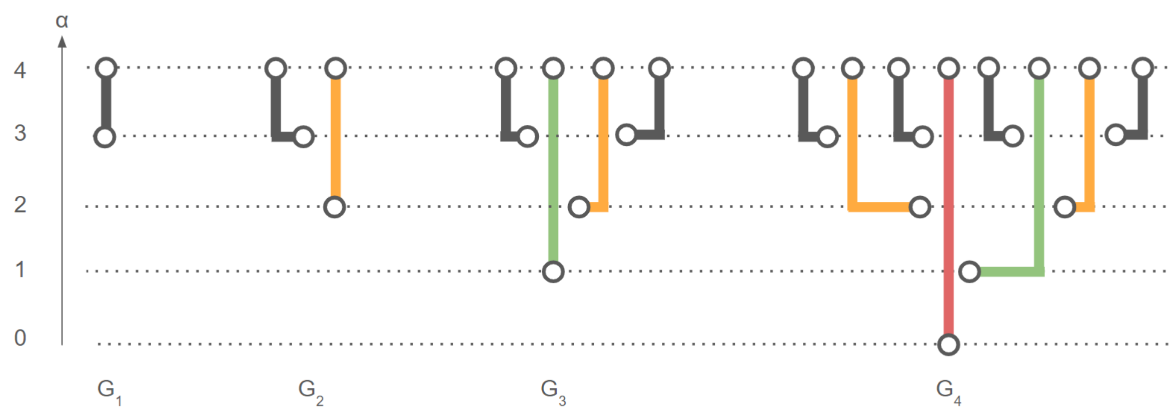

Figure 15: Symmetry groups detected by the existing symmetry detection method

Group	Descriptor (h)	Occurrences
G_1	(3, 4, < >)	4
G_2	(2, 4, <1>)	2
G_3	(1, 4, <1, 2>)	1
G_4	(0, 4, <1, 2, 3>)	1

	G1	G2	G3	G4
G1	1	0.5	0.25	0.125
G2	—	1	0.6	0.333
G3	—	—	1	0.636
G4	—	—	—	1

Figure 16: Left: Hierarchy descriptors and occurrences of each found group. Right: Symmetric matrix containing the symmetry scores S of all pairs of group descriptors

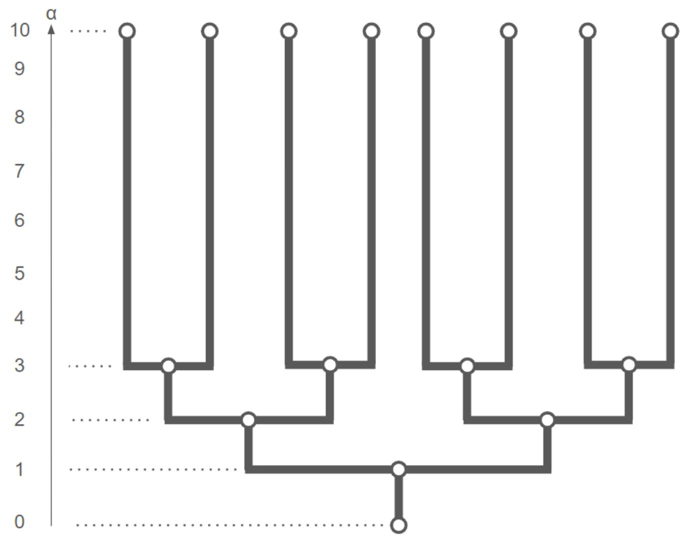

Figure 17: Scaled version of the base case tree with leaf scalar values set to 10.

Group	Descriptor (h)	Occurrences
G_1	(3, 10, < >)	4
G_2	(2, 10, <1>)	2
G_3	(1, 10, <1, 2>)	1
G_4	(0, 10, <1, 2, 3>)	1

	G1	G2	G3	G4
G1	1	0.636	0.368	0.2
G2	—	1	0.652	0.384
G3	—	—	1	0.659
G4	—	—	—	1

Figure 18: Left: Group descriptors and occurrences of branches in groups G_i for the scaled base case with leaf scalar value 10. Right: Matrix with symmetry scores between for the pairs of group descriptors.

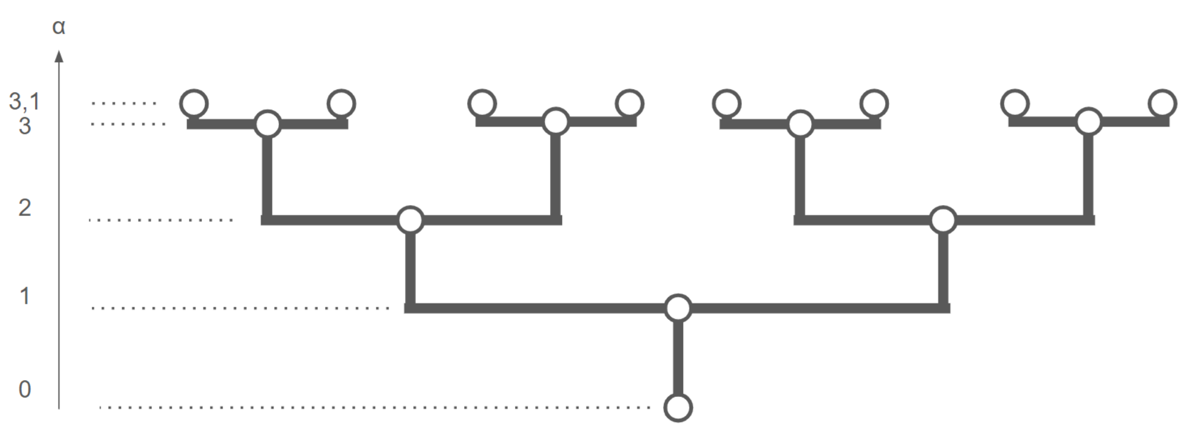

Figure 19: Scaled version of the base case tree with leaf scalar values set to 3.1.

Group	Descriptor (h)	Occurrences
G_1	(3, 3.1, < >)	4
G_2	(2, 3.1, <1>)	2
G_3	(1, 3.1, <1, 2>)	1
G_4	(0, 3.1, <1, 2, 3>)	1

	G1	G2	G3	G4
G1	1	0.153	0.057	0.025
G2	—	1	0.521	0.266
G3	—	—	1	0.607
G4	—	—	—	1

Figure 20: Group descriptors and occurrences of branches in groups G_i for the scaled base case with leaf scalar value 3.1. Right: Matrix with symmetry scores between the pairs of group descriptors.

7 The connection between symmetry groups and larger material structures

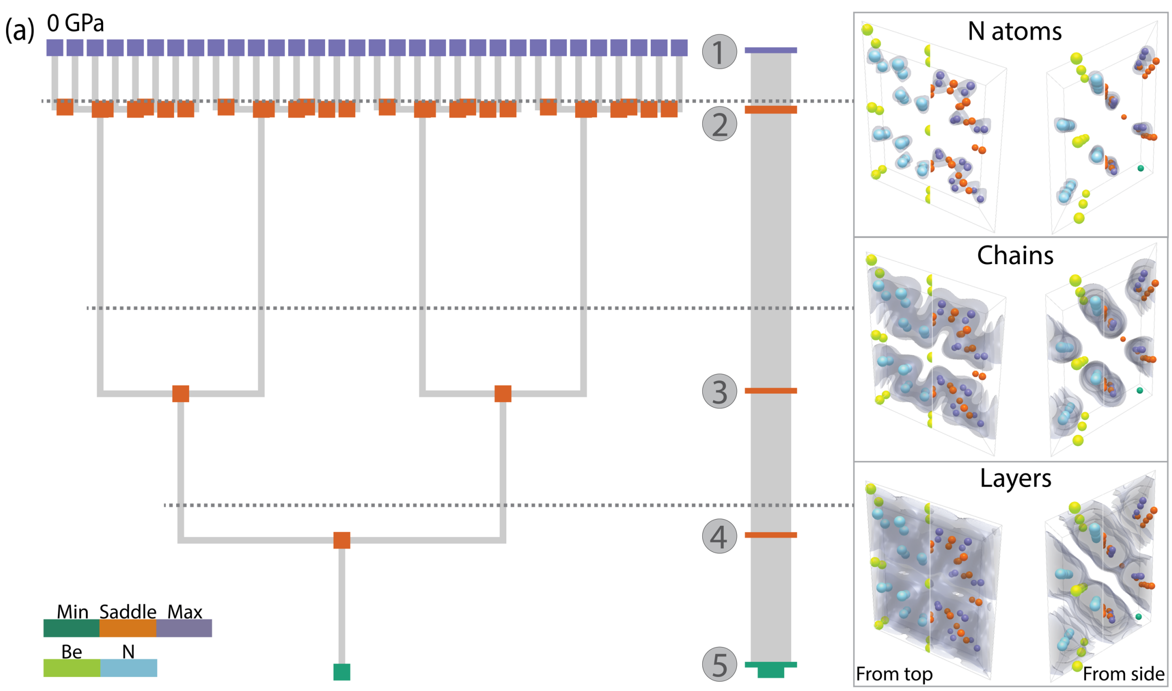

Figure 21: Figure from the work by Thygesen et al [TAS+23] showing the correlation between distinct layers of saddle points in the merge tree and specific geometrical structures in the corresponding level sets for the crystal BeN4. Left: The merge tree. Middle: Collapsed bar representation of the merge tree. Right: Corresponding level set representation at a given charge density value α

In highly symmetrical crystals such as BeN4 in figure 21 there is a clear connection between symmetry in the merge tree and larger structures such as chains and layers. In the figure it can clearly be seen that the crystal consists of individual atoms in the interval between node layers 1 and 2, geometrical chains between node layer 2 and 3 and then geometrical layers between the node layers 3 and 4. However, in the presence of asymmetry, as seen in figure 9, these types of node layers are less detectable or possibly non-existent.

While a connection between node layers in merge trees and specific geometrical structures exists, there are several more factors to consider when trying to detect these structures via the merge tree.

Even though distinct layers of saddles exist in the merge tree, it is not possible to know explicitly from the merge tree what sort of structure the layer represents geometrically. While a layer for example could represent the transition from a chain structure to a layer structure, it could just as well represent the transition from short chain structures into longer chains. So while it is possible to identify a transition between structures, it is not possible to identify the type of structure and whether the transition actually changes the dimensionality of the structure or if it just expands in the same dimension. This knowledge is a geometrical property which can not be deduced from the merge tree since it is a purely topological representation. Similarly, several degrees of structure transitions can happen almost instantaneously depending on the material. In the merge tree, this corresponds to several node layers representing geometrical transfers being very close to each other or even overlapping. Beside the interpretation of identified layers, one also has to consider cases when there are no clearly distinguishable layers at all.

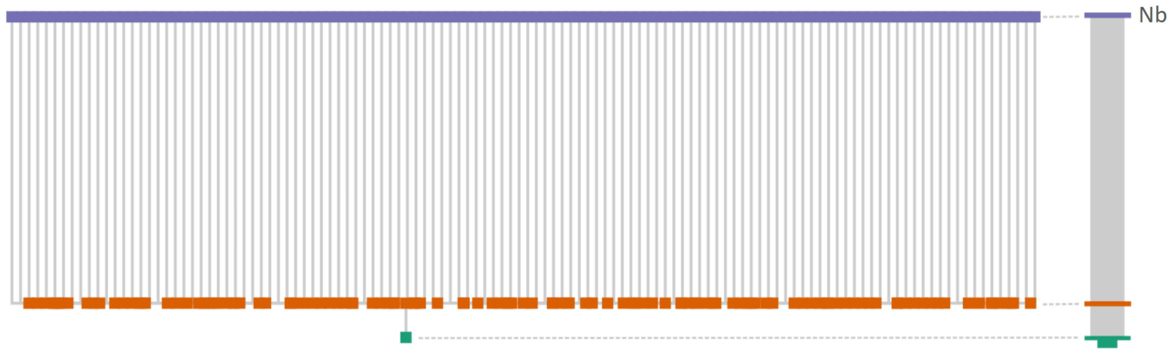

Figure 22: Merge tree and corresponding collapsed bar representation of the metal niobium (Nb). Leaf nodes are shown in blue, saddles point in orange and the root node in green. Used with courtesy of Signe Sidwall Thygesen

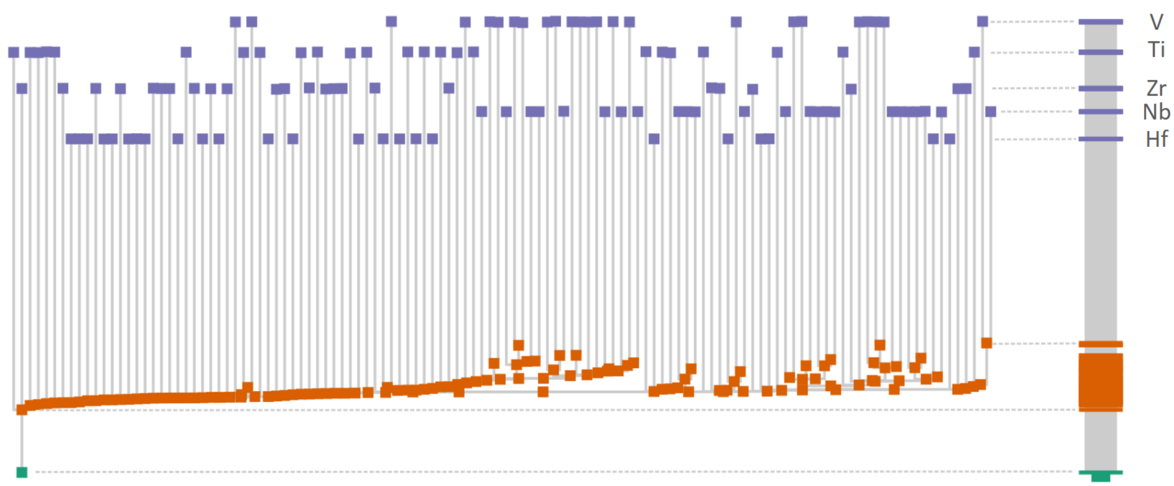

Figure 23: Merge tree and corresponding collapsed bar representation of a niobium-based alloy containing vanadium (V), titanium (Ti), zirconium (Zr), niobium (Nb) and hafnium (Hf). Each layer of leaf nodes (blue) corresponds to a specific type of atom. Saddle points are shown in orange and the root node in green. Used with courtesy of Signe Sidwall Thygesen

Similarly to crystals, pure metals also have a high degree of symmetry and clear layering of nodes in their merge tree representation as seen for the metal niobium in figure 22. However, when other materials are introduced — either by mixing metals into alloys or by adding impurities to the crystals

— the influence of the added elements can affect the previously symmetric and layered node placements in the merge tree. Figure 23 shows a merge tree and corresponding bar representation of a niobium-based alloy with additional elements. While the leaf nodes still form clear layers based on atom type, the previously compact layer of saddle points have become spread out due to the influence of the new elements. In cases like this where the transition happens gradually over a wider scalar interval it is not possible to distinguish whether it is only one type of geometrical transition happening in the interval or several different ones. Some subsets of the tree might have distinct partial transitions and clear layers in the interval, while other subsets of the tree do not follow the same pattern. There also exist cases with such high levels of structural asymmetry that their corresponding merge trees lack any visual or structural indication of saddle point layers at all.

8 Discussion and future work

8.1 Further definition of symmetry in tree structures with regards to axis

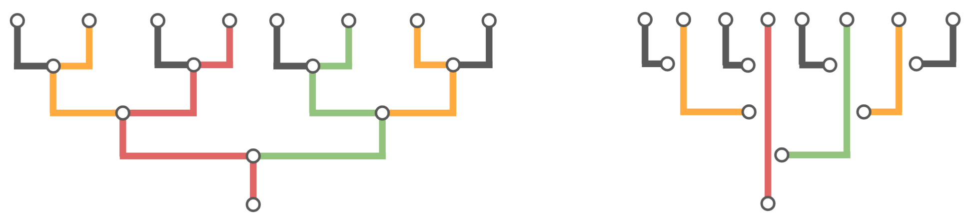

Figure 24: Visualization of how branch decomposition affects the ideal base case tree

The derived symmetry descriptors h in figure 15 shows that the structure of groups generated by the suggested algorithm is the same as the expected groups structures visualized in figure 12. However, if we compare the number of expected occurrences in figure 13 and the actual generated occurrences in figure 16 there is a difference which implies that the method suggested by Thomas and Natarajan does not fulfill the previously defined symmetry criteria for the given context. By visualizing the brach decomposition of the base case tree in figure 24 it is clearly shown that the problem lies in the structural influence of the branch decomposition. In the case of the ideal base case tree, half of the top edges which are expected to belong to group G_1 are instead merged with their parents since they are classified as part of a longer path according to branch decomposition. This in turn breaks their symmetry with other edges on the same height. In other words, the problem is the merging of edges with regards to longest path and using the resulting branches as an axis of symmetry for the surrounding edges.

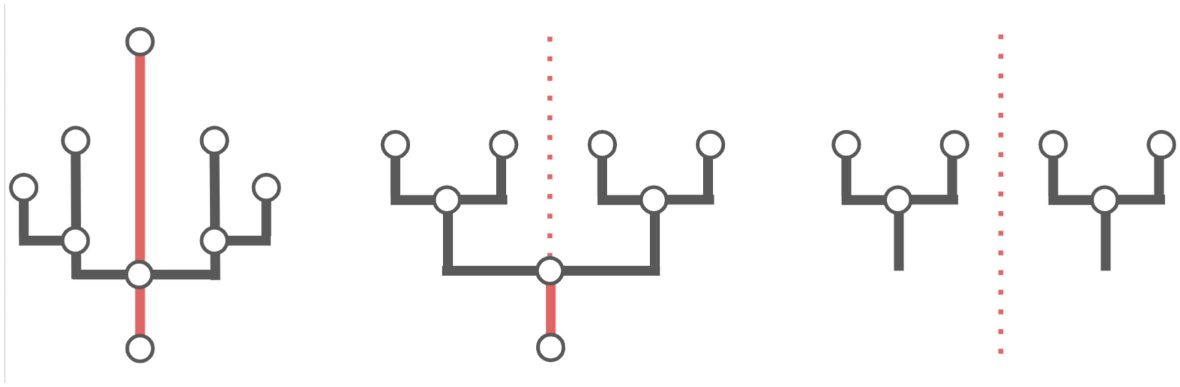

Figure 25: Illustration of the different perspectives on symmetry axes with the axis of symmetry in red. Left: Explicit axis perspective, middle: semi-implicit axis perspective and right: implicit axis perspective.

To further understand and work around this problem, three different perspectives on symmetry axes for data trees were defined as visualized in figure 25. Namely, explicit axis perspective, semi-implicit axis perspective and implicit axis perspective.

For explicit axis perspective, subsets of branches in the tree are divided by using an explicit path within the tree as an axis of partition. This makes the said path a structural axis of symmetry for its surrounding subsets — even though the actual symmetry might depend on some external parameter — and it restricts which subsets can be derived from the tree structure. This is the approach used in branch decomposition to derive the branches.

Implicit axis refers to when the axis of symmetry is an entirely external parameter or structure and not an explicit path within the given tree structure. In the given context, the charge density scalar axis would act as the symmetry axis. This means that any arbitrary subset of connected branches can be symmetrically compared with any other subset, without being restricted by their relation to each other or specific structures in the tree. An important notion is that this approach do not define to how the subsets are divided, so no direct consideration is taken into account about the structural aspects of the data.

The semi-implicit axis perspective is a middle ground between the two other perspectives. Instead of using a full path as a divider for the subsets, the subsets are created by splitting the tree directly at the saddle points, which makes the previous edge a local explicit axis of symmetry. While the explicit axis perspective can be seen as a vertical partition of subsets around a prioritized path, this approach partitions subsets horizontally by treating both edges around a saddle point as equal. It further specifies that a subset in this perspective contains all children to the highest parent node in the subset, unlike with the implicit axis perspective which do not define the partition of subsets at all.

Depending on the given symmetry conditions and their complexity, the implicit axis can be any type of parameter or n-dimensional surface, as long as its values can be mapped to the merge tree to enable similarity comparisons around it. Tree structures whose symmetry is defined horizontally by the semi-implicit or implicit perspective will from here on be referred to as *flat hierarchies* since they do not contain or depend on explicit main stem structures in the data. Flat hierarchies encompass that all edges and subsets of edges on the same height in the tree are treated as equal in terms of symmetry. The material data in the given context is a flat hierarchy since the structure is infinitely repeating — meaning there is no single most important path — and all edges are equally important in terms of symmetry, independent of which scalar intervals they appear at.

8.2 The dangers of defining subsets in flat tree hierarchies with regards explicit axis perspective

As seen in figure 24 and explained in section 8.1, applying subset segmentation in flat hierarchies with regards to explicit axis perspective forcibly creates artificial main axes where there originally were none. By merging edges into longer paths, the the edges lose their symmetry to other surrounding edges on the same height which previously might have been fully symmetrical.

This segmentation approach also affects the symmetry scores for scaled trees. Since the branches are defined vertically, the symmetry scores — which are based on horizontal similarity around the axis — scales disproportionally when a horizontal layer of edges is scaled as in figure 17 and 19. Their symmetry scores are shown in figure 20 and 18. Different atom types have different charge densities which results in a natural scaling of node layers in the corresponding merge tree representation. This means that fully symmetrical materials will get vastly different symmetry scores with this approach depending on which atoms they contain. An explicit axis approach of segmentation, such as branch decomposition, is furthermore ill-suited for tree structures which contain several leaf nodes with the same scalar value. In these cases, the longest paths in the tree will be chosen at random depending on which node is chosen as the highest by the decomposition schema. In the ideal symmetric case in figure 11 the branches of the merge tree will have the same structure regardless of which path is chosen. But for any other non-ideal merge tree, different path choices will yield different variants to how branches are segmented and scored with regard to symmetry. One path might result in completely different symmetry groups than another path in the same tree. Applying explicit axis-based symmetry approaches to flat hierarchies will therefore result in unpredictable segmentation and classification of subsets. And while some unit cells can contain a single highest leaf node which might seem like a natural axis of symmetry, this does not represent the physical reality since the unit cell is a periodical subset of the material which repeats infinitely. When expanded, the material and its corresponding merge tree have no distinct longest path in accordance to a flat hierarchy.

8.3 Alternative approach to symmetry detection for flat hierarchies

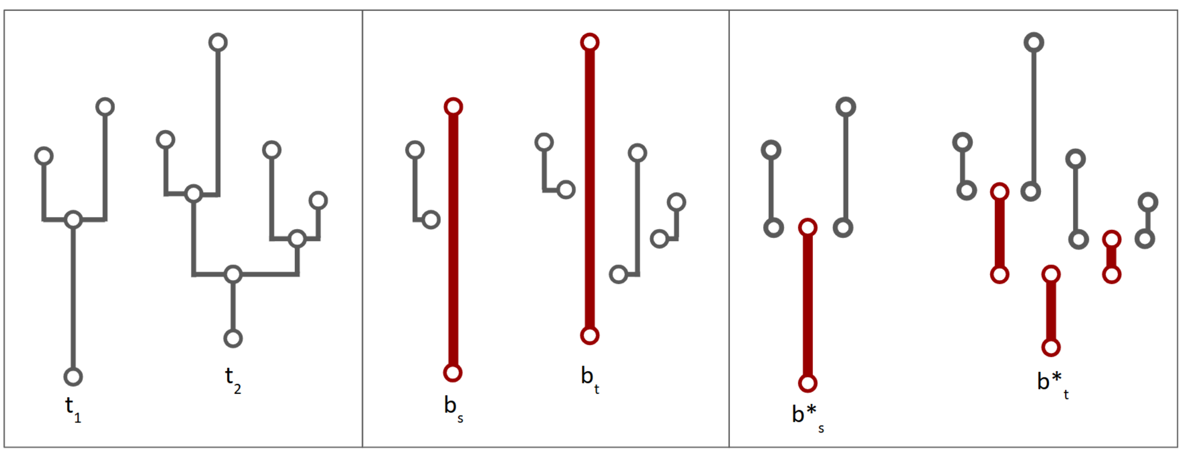

Figure 26: Two arbitrary sub-trees in a merge tree (left), the same sub-trees after applying branch decomposition (middle) and the sub-trees after splicing them with regards to a semi-implicit axis perspective. The local axes of symmetry are shown in red.

Since the existing method for symmetry detection in merge trees is based on explicit axis perspective and therefore not compatible with the given symmetry criteria, an alternative method for symmetry detection is suggested for the given context. This method is a modified version of the method described in section 2.4 which instead bases the segmentation of branches on semi-implicit axis perspective. Figure 26 visualize how this looks in practice for two arbitrary branches of a merge tree compared to the explicit axis approach. The red edges represent a local structural axis of symmetry around which the surrounding branches are defined.

$$S_{mod} = \frac{O_{max} + k}{p_s + p_t} \cdot 100 \tag{2}$$

Since flat hierarchies do not have a prioritized main paths, the edges which makes up the local axes can be treated as any other edge in the branch subset. Therefore the modified symmetry score S_{mod} in equation 2 only contain a single overlap value O_{max}. This is the maximum assignment solution for branch overlap in the two compared subsets with regards to the scalar value α. Although, this score can be misrepresenting depending on combinations of factors such as the total height or amount of edges in each branch, or further external factors such as atom type or other physical attributes. Therefore a balancing factor k is suggested as well which represents the relation between these potential biases.

Similarly to the original method, branches are defined in terms of hierarchy descriptors h and group descriptors G to make the representation more compact. But instead of assigning the shortest and utmost branches to groups first, as with the explicit axis based-approach — this method instead process the highest located edges first in a top-down manner from leaf to root to ensure that all internal symmetry groups within a larger group are defined before the larger group is defined and assigned with regards to the smaller internal groups.

8.4 Detection of larger geometrical structures in level sets via the merge tree representation

As explained in section 7 it is not possible to identify strictly geometrical properties in isosurfaces explicitly from the merge tree since it is a topological representation. However, it is shown in figure 21 that uniform topological transitions between different geometrical structures in isosurfaces corresponds to layers of nodes in the merge tree. This property of the merge tree could therefore be used to partially detect specific transitions or intervals of interest in a level set for later geometrical analysis.

The collapsed bar representation of the merge tree can be viewed as the distribution of nodes along the charge density α axis. If there are clear distinguishable groups of nodes in the bar which only span a small interval, such as in figure 21, then the group is likely an indicator of a structural transition. By extracting these scalar intervals of interest, either algorithmically or visually via the collapsed bar representation, geometrical analysis through additional methods can be applied more efficiently on the corresponding level sets. For a visual approach of extraction it is however not currently possible to differentiate if a line in the bar represents only a single node — which would be considered an outlier in this context — or an entire group of nodes at the same scalar value. For that reason, further visual indicators need to be applied in order to visualize the amount of overlapping nodes in the bar representation.

8.5 Comparison of entire trees

A possibility with the current implementation and processor design is that it could enable compact comparison between entire merge trees. Since doped crystal materials to some degree keep the hierarchical structure and symmetry from its initial host material as seen in figure 7 and 9, it could be possible to save the detected symmetry groups from a initial host material and then use them as a ready-made group template for its polluted variants. If enough similarity is maintained between the merge trees, the symmetry detection schema should only have to create new symmetry groups for the branches corresponding to impurities. This could significantly shorten the calculation time with regards to cross-comparisons. But as mentioned in section 3, the influence of an impurity can create global scalar displacements in the unit cell of the doped materials compared to the pure host material. This makes this approach harder or possibly not feasible to implement in practice unless there exists an efficient way to map and rescale the group descriptors with regards to this scalar displacement. This mapping factor then needs to be considered in the symmetry detection schema since it affects both the group assignments within the trees, as well as the similarity between different trees.

8.6 Comparison of static node attributes

An addition to the comparison of scalar overlap to determine symmetry between branches, it would also be of importance to add comparison of static node attributes, such as atom types. By incorporating atom types into the leaf nodes and then use type matching as an initial boundary condition, the number of cross-comparisons would go down significantly. It would further make the symmetry groups better represent reality since the current implementation only consider topological changes, without regards to what the nodes and level sets actually represents in the physical domain. Similar types of initial comparisons can be done for any static node or edge attribute relevant to symmetry to reduce the need for cross-comparison and calculation of overlaps.

8.7 Testing of the suggested method

To test the suggested method it should be applied on the base case 11 in the same way as the previous method. It should also be applied on variants of the base case with isolated additional characteristics to test different aspects of robustness. These cases should at least include addition of noise, large outlier nodes, partial overlaps, singular outliers and saddle hierarchies which are considered unstable with regards to distance. For the sake of controllability, it is suggested that testing on real data is not done until sufficient results are achieved on the ideal synthetic base case variants.

8.8 Future work

The natural continuation of this project is to implement and test the suggested method to see if it yields better results than the previously tested one. It would also be of interest to evaluate possible optimization of the comparison algorithm and incorporate external attribute comparisons such as atom type, spin and magnetization to the symmetry detection to further understand the properties of the data. Another aspect of interest is the visual representation of the merge tree and how to best incorporate the symmetry groups in an interactive way to enable exploration and understanding.

8.9 Conclusion

Based on the results in section 6 and the reasoning in section 7, the structural symmetry in a merge tree representation of a charge density field is defined by the following factors:

- The scalar overlap of individual branches edges in each sub-tree.
- The amount of branches in each sub-tree.
- Implicit symmetry axes

The branch decomposition-based approach to detect symmetry in the merge tree is not applicable to the given context of material analysis since it uses an explicit axis approach which creates artificial main branches and destroys flat hierarchies.

Regarding detection of geometrical structures it is only partially possible to detect specific crystal structures explicitly from the merge tree. The degree to which detection is possible further varies depending on the material. But in the collapsed bar form, these partial indication could be used to detect scalar intervals of interest which are more likely to contain potential structure transitions. These could then later be combined with further geometrical methods to identify the actual specific structures and transition values in the charge density.

References

[DBTA23] J. Davidsson, B. Bertoldo, K.S. Thygesen, and R. Armiento. Absorption versus adsorption: high-throughput computation of impurities in 2d materials. *npj 2D Materials and Applications*, 7(26), 2023.

[JGJGC18] Tierny J., Favelier G., Levine J.A., and Michaux M. Gueunet C. The topology toolkit. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):832–842, 2018.

[Mun57] J. Munkres. Algorithms for the assignment and transiortation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[SSK+15] Erik Sunden, Peter Steneteg, Sathish Kottravel, Daniel Jonsson, Rickard Englund, Martin Falk, and Timo Ropinski. Inviwo - an extensible, multi-purpose visualization framework. In *2015 IEEE Scientific Visualization Conference (SciVis)*, pages 163–164, 2015.

[SSW14] H. Saikia, H. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum*, 33(3):41–50, 2014.

[TAS+23] S.S. Thygesen, A.I. Abrikosov, P. Steneteg, T.B. Masood, and I. Hotz. Level of detail visual analysis of structures in solid-state materials. 2023.

[TN11] D.M. Thomas, , and V. Natarajan. Symmetry in scalar field topology. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 17(12), December 2011.

[YMS+21] L. Yan, T.B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Computer Graphics Forum*, 40(3):599–633, 2021.