TẠP CHÍ KHOA HỌC
TRƯỜNG ĐẠI HỌC QUY NHƠN

Xây dựng cấu trúc phân tử bằng thuật toán kết hợp K-Nearest Neighbor và cây tìm kiếm K-Dimension

Trương Thị Cẩm Mai[1],*, Nguyễn Trương Thành Hưng[2,3]

[1]Khoa Khoa học Tự nhiên, Trường Đại học Quy Nhơn, Việt Nam
[2]Quy Nhơn AI, FPT Software, Việt Nam
[3]Khoa Kỹ thuật, Trường Đại học Friedrich-Alexander Erlangen-Nürnberg, Đức

Ngày nhận bài: 23/01/2022; Ngày sửa bài: 02/05/2022;
Ngày nhận đăng: 14/05/2022; Ngày xuất bản: 28/06/2022

TÓM TẮT

Việc xây dựng các đặc tính và cấu trúc phân tử đóng một vai trò quan trọng trong nhiều lĩnh vực khác nhau, như khoa học vật liệu, cảm biến, công nghệ nano, thiết kế và khám phá thuốc. Tuy nhiên, việc xây dựng cấu trúc phân tử trên một tập dữ liệu thô, tập dữ liệu bị nhiễu và thiếu thông tin, là một nhiệm vụ đầy thách thức nhưng rất quan trọng. Thuật toán phân loại K-Nearest Neighbors (KNN) là một thuật toán lazy learning, có xu hướng tìm kiếm các điểm gần nhất cho một mục tiêu trong toàn bộ tập huấn luyện. Tuy nhiên, quá trình dự đoán của KNN khá mất thời gian. Trong khi thuật toán cây tìm kiếm K-Dimension (K-D tree) là một cây nhị phân đa chiều, có cấu trúc lưu trữ cụ thể để biểu diễn dữ liệu huấn luyện một cách hiệu quả về mặt thời gian. Từ các khía cạnh trên, trong bài báo này, chúng tôi đã thử nghiệm và đề xuất một phương pháp gọi là thuật toán cây tìm kiếm KNN-KD để xử lý tập dữ liệu thô về cấu trúc phân tử bằng cách kết hợp các ưu điểm của KNN và cây K-D.

Từ khóa: *Xây dựng cấu trúc phân tử, học máy, cây tìm kiếm K-Dimension, K-Nearest neighbors.*

*Tác giả liên hệ chính.
Email: truongcammai@qnu.edu.vn

https://doi.org/10.52111/qnjs.2022.16305

Construction of molecular structures using K-Nearest Neighbor with K-Dimension tree algorithm

Truong Thi Cam Mai[1,*], **Nguyen Truong Thanh Hung**[2,3]

[1]*Faculty of Natural Sciences, Quy Nhon University, Vietnam*
[2]*Quy Nhon AI, FPT Software, Vietnam*
[3]*Faculty of Engineer, Friedrich-Alexander University Erlangen-Nürnberg, Germany*

Received: 23/01/2022; Revised: 02/05/2022;
Accepted: 14/05/2022; Published: 28/06/2022

ABSTRACT

The construction of molecular properties plays a significant role in various fields, such as material science, sensors, nanotechnology, drug design, and more. However, the construction of molecular structures on a raw dataset, which is noisy and incomplete, is a challenging but crucial task. K-Nearest neighbor Classification (KNN) is a lazy learning classification algorithm with tendency to search the nearest neighbors for a target in the entire training set. Nevertheless, each step of KNN is quite time-consuming. In comparison, the K-Dimension tree (K-D tree) algorithm is a multi-dimensional binary tree, a specific storage structure for time-efficiently representing training data. To that respect, in this journal article, we conduct and propose a method called the KNN-KD tree algorithm to process a raw labeled dataset of the molecular properties by combining the advantages of the KNN and K-D tree.

Keywords: *Construction of molecular structures, machine learning, K-Dimension tree, K-Nearest neighbors.*

1. INTRODUCTION

The construction of molecular structures is one of the widespread issues where various approaches are applied using traditional chemistry formulae or mathematic computations. However, the datasets collected in experiments are noisy and incomplete for reconstructing molecular structures. In this article, a raw chemical dataset of Chemistry and Mathematics in Phase Space (CHAMPS)[1] is used to prove the performance of the geometric-based approximated machine learning model, namely the K-Dimension tree (K-D tree) in the construction of molecular structures. Significantly, this journal article will analyze, construct and visualize the molecular structures while we only use the XYZ coordinates of atoms for training the model. Consequently, this method can reduce the computing time with comparably high accuracy to rule-based methods.

1.1. Construction of molecular structures

The construction of molecular structures is a typical issue in chemistry since it impacts biomedical engineering, drug discovery, and vaccine exploration. In the real scenario, much information on the molecular properties is noisy, incomplete, and deficient.[2] As a demand, we need methods that improve the exploration of the molecular structures to deal with the shortage of information.

**Corresponding author.*
Email: truongcammai@qnu.edu.vn

https://doi.org/10.52111/qnjs.2022.16305

Various approaches to constructing the molecular structures are applied with machine learning.[3,4] However, the complexity of constructing the molecular structures increase significantly once the bonding schema is involved.[3] One of the state-of-the-art methods which achieve high accuracy on the CHAMPS dataset is a hybrid approach, namely soft graph transformer by Bosch Corporate Research and Bosch Center for AI.[5] This model processes the entire molecule one by one, simultaneously predicting each of the scalar couplings in the molecule. Instead of using a traditional graph model, their approach processes the data as a meta-graph where each atom, chemical and non-chemical bonds, namely just pairs of atoms, are included in the model, and even triplets or quads all become nodes for the graph transformer. Distance measurement between all the nodes in the graph is necessarily defined to support the model. For example, atom-to-atom distances use the actual distance between atoms. In contrast, atom-to-bond distances use the minimum distance from the atom to the two atoms in the bond, with similar extensions for triplets quads. Some other methods apply the Bidirectional Encoder Representations from Transformers (BERT) training to extract only raw coordinates instead of distance, translational and rotational invariances, such as MTM[6], Mol-BERT,[7] BERT of Xin-Yu et al.[8]

Most other studies on molecular structures based on the CHAMP dataset have focused on predicting the scalar coupling constant without having a proper way to process and classify the bond information. Hence, the success rate of bond reconstruction is not good enough for further steps. Moreover, the running time of extracting information from coordinate files is time-consuming due to the enormous-size dataset. Also, some approaches applying multiprocessing may yield incorrect data. For the above reasons, a high percentage of molecule structures are not constructed correctly, which leads to the fact that various models cannot improve the accuracy in predicting the scalar coupling constants. Consequently, there is a need to create an easily customed algorithm to improve the success rate of the construction of molecular structures.

1.2. K-Nearest neighbors algorithm

K-Nearest Neighbors (KNN) algorithm is based on the distance metric function, namely Euclidean distance, to calculate the distance between the sample to be classified x and each sample in the training set, sort the calculated distance, and select the k training samples closest to the sample to be classified as the k nearest neighbors of x. If the sample belonging to a particular class of the k nearest neighbors is the majority, the representative classification sample x is classified into the category.[9]

1.3. K-D tree algorithm

K-Dimension tree (K-D tree) is a binary tree structure that recursively partitions the parameter space along the data axes, splitting it into nested orthotropic regions into which data points are filed.[10] K-D tree is a particular case of binary space partition trees. In detail, it is a space partitioning data structure for organizing points in a K-Dimensional space. A non-leaf node in the K-D tree divides the space into two parts, called half-spaces. Each subspace can be recursively divided in the same way. The left subtree of that node represents points to the left of this space, and the right subtree represents points to the right of the space. Constructing a K-D tree on a K-Dimension dataset represents a partition of the K-Dimensional space formed by the K-Dimensional dataset.

K-D trees are helpful in range searches and nearest neighbor searches. They are the most powerful data structures for small and moderate numbers of dimensions up to 20 dimensions.[11] In general, structures of the K-D tree attempt to reduce the required number of distance calculations by efficiently encoding aggregate distance information for the sample.

The construction of a K-D tree is speedy since partitioning is conducted only along the data axes. Once built, the nearest neighbor of a query point can be determined with only distance computation. Nevertheless, the K-D tree method is high-speed for low-dimensional neighbor searches. It becomes ineffective as it grows tremendous. The primary reason is that the ratio of the volume of a unit sphere in K-dimensions falls exponentially compared to a unit cube in K-dimensions. Thus at an exponential rate, many cells have to be searched within a particular radius of a query point, say for a nearest-neighbor search. Additionally, the number of neighbors for any cell grows up and eventually becomes insurmountable.[12]

1.4. Improved neighbor search algorithm using a K-D tree to find multiple k nearest neighbors (KNN-KD tree)

However, the nearest neighbor searching algorithm applied with the original K-D tree can only find one nearest neighbor. Consequently, it is necessary to adapt the original algorithm to be more efficient for searching the molecular data called KNN applied with the K-D tree (KNN-KD tree).[13,14] It can discover multiple K-nearest neighbors of a given query point instead of just finding one nearest neighbor. A bounded priority queue that stores the list of K-nearest neighbors together with their distances to the query point is applied in the adapted algorithm. The higher the priority value of the point is, the longer the distance from that point to the query point becomes. A fixed upper bound of the bounded priority queue must be defined, which is the number of nearest neighbors. The bound is used to prune tree searches, so if a series of K-nearest neighbor queries are required, it may help supply the distance to the nearest neighbor of the most recent point. Whenever a new point is added to the queue, if the queue is at capacity, the point with the longest distance to the query point is ejected from the queue.[15]

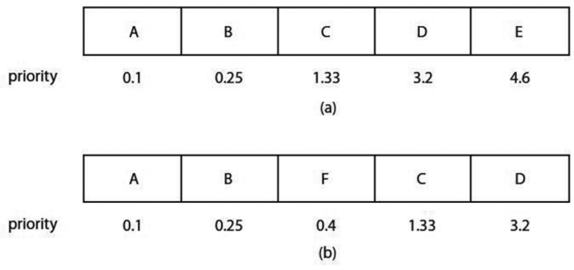

Figure 1. Bounded priority queue for KNN.

For example, Figure 1a shows the nearest neighbor priority queue with the upper-bounded size of five and holds five points, from A to E. Suppose that the next nearest neighbor point to be inserted into the priority queue is the point F with the priority of 0.4. Because the maximum size of the priority queue is five, point F is inserted into the priority queue. However, point E with the longest distance to the query point q is eliminated. Figure 1b shows the resulting priority queue after point F is inserted. On the other hand, suppose that the next nearest neighbor to be inserted into the priority queue is point G with a distance of 3.5. Because the distance value of G is greater than the maximum priority element in the queue, G is not inserted into the queue.

In conclusion, there are two improvements in the KNN-KD tree algorithm from the traditional K-D tree to improve search efficiency. The first improvement is that when determining whether to look on the opposite side of the splitting hyperplane, the algorithm applies the distance from the point with the longest distance in the nearest neighbor priority queue as the radius of the candidate hypersphere.[16] The second improvement is that it reduces the time complexity from $O(n)$ to $O(n^{1-1/k}+m)$.[17]

2. DATASET AND RESEARCH METHOD

2.1. Dataset

Because the training and test splits are by molecule, no molecule in the training data will be found in the test data. The dataset contains these files as follows.

- **train.csv** The training dataset contained 4,658,147 scalar coupling observations of 85,003 unique molecules. The first column (*molecule_name*) is the molecule's name where the coupling constant originates. The second (*atom_index_0*) and the third column (*atom_index_1*) are the atom indices of the atom pair, creating the coupling. The fourth column (*scalar_coupling_constant*) is the scalar coupling constant needed for predicting. All of the molecules contained five types of atoms: carbon (C), hydrogen (H), nitrogen (N), fluorine (F), and oxygen (O). There were eight distinct types of scalar coupling, including 1JHC, 1JHN, 2JHH, 2JHC, 2JHN, 3JHH, 3JHC, and 3JHH, which means that the fluorine coupling is not presented in the dataset.

- **test.csv** The test set has the same information as the train set but without the target variable, namely the scalar coupling constant. Because scalar coupling constant contains information about relative bond distances and angles, which are informative in determining the connectivity between atoms in a molecule, scalar coupling constant is not available in the test dataset to evaluate whether the model is robust to it.[18] The test dataset contained 2,505,542 scalar coupling observations of 45,772 unique molecules.

- **structures.csv** contains the molecular structure XYZ information, where the first column (*molecule_name*) is the molecule's name, followed by the index of the atom *(atom_index)*. The following column (*atom*) contains the atomic element symbols such as H for hydrogen, C for carbon, N for Nitrogen. The remaining columns include the *X*, *Y,* and *Z* cartesian coordinates.

- **dipole_moments.csv** contains the molecular electric dipole moments. These are three-dimensional vectors that indicate the charge distribution in the molecule. The first column *(molecule_name)* are molecule's names; the second to the fourth column is the XYZ components of the dipole moment.

- **magnetic_shielding_tensors.csv** contains the magnetic shielding tensors for atoms in the molecules. The first column (*molecule_name*) is the molecule name, the second column (*atom_index*) is the index of the atom in a molecule, the third to eleventh columns comprise the XX, YX, ZX, XY, YY, ZY, XZ, YZ, and ZZ elements of the tensor/matrix respectively.

- **scalar_coupling_contributions.csv** The scalar coupling constants in the train set (or corresponding files) are a sum of four terms. *scalar_coupling_contributions.csv* contains all these terms. The first column (*molecule_name*) is the name of the molecule. The second (*atom_index_0*) and third column (*atom_index_1*) are the atom indices of the atom pair. The fourth column shows the type of coupling. The fifth column (*fc*) is the Fermi Contact contribution. The sixth column (*sd*) is the Spin-dipolar contribution. The seventh column (*pso*) is the Paramagnetic spin-orbit contribution. Finally, the eighth column (*dso*) is the Diamagnetic spin-orbit contribution.

2.2. Proposed method

This journal article proposes to apply the K-Nearest Neighbour with the K-D tree (KNN-KD tree) algorithm to solve the construction of molecular structures, where the knowledge of geometry and pattern matching is utilized.

Our KNN-KD tree algorithm is split into four main steps to reconstruct the molecular structures based on the bonding schema. Every molecule is selected and restored by four steps where the XYZ cartesian coordinate of atoms structure is applied in the K-D tree. Our method solves the three kinds of coupling types.

In general, the bond length between the two atoms is approximately the sum of the covalent radii of the two atoms. Consequently, in the bonding reconstruction algorithm, the valence radii of the chemical elements are pre-defined and applied. The covalent radius is the distance from the center of the nucleus to the outermost shell of the electron, and its value may be derived from experimental measurements or calculated by theoretical models.[6] However, these relationships are certainly not accurate due to the inconstant size of an atom but depend on its chemical environment.[19] For example, in the heteroatomic A-B bonds, ionic terms may enter.[20] Furthermore, the differential value of single, double, and triple bonds are too small to distinguish based on the distance values derived from the XYZ cartesian coordinate of atoms in the dataset. Consequently, in our algorithm, only the single bond covalent radius is manipulated to create the general bonding schema.

2.2.1. Single bond connect reconstruction

To successfully reconstruct the total bonding system, firstly, the overall molecular bonding structure must be created. The background of single bond connect reconstruction is based on the bond length comparison. In detail, each time, a pair of XYZ cartesian coordinates of two atoms in a specific molecule is put into the K-D tree structure in the three-dimensional geometry. To be more easily understandable, a random molecule, *dsgdb9nsd_000007*, is analyzed.

Table 1. The XYZ cartesian coordinate of the molecule *dsgdb9nsd_000007*

index	atom	x	y	z
0	C	-0.0187	1.5256	0.0104
1	C	0.0021	-0.0039	0.0019
2	H	0.9949	1.9397	0.0029
3	H	-0.5421	1.9236	-0.8651
4	H	-0.5252	1.9142	0.9000
5	H	0.5255	-0.4019	0.8775
6	H	-1.0115	-0.418	0.0095
7	H	0.5086	-0.3924	-0.8876

Table 1 shows that the molecule *dsgdb9nsd_000007* has eight atoms inside, which is two carbon atom and six hydrogen atoms. With the XYZ cartesian coordinate, it is easy to outline the position of every atom inside the molecule in the three-dimensional geometry, as illustrated in Figure 2. Still, there is no connection between these atoms in the dataset, namely the bonding schema. The approach is to search for all possible connections to create single chemical bondings from an atom to others. Since the valence of each atom is not provided, the standard valence is used as the default value. Also, any atom with zero available bonding is rejected. With the molecule *dsgdb9nsd_000007*, the algorithm starts with the first atom of hydrogen because the processing order beginning with the hydrogen avoids a butadiene-like molecule.

Based on the K-D tree query, the nearest atom is chosen by distance. At first, the K-D tree finds the nearest atom to a selected atom. For the spatiotemporal interpolation for the *dsgdb9nsd_000007* data, a three-dimensional K-D tree has been constructed to find the K-nearest neighbors. Figure 3 illustrates the K-D tree built from eight atomic points of *dsgdb9nsd_000007* data alongside its index. Atom C at *i=1* is the root point because it is the median point on the x-axis and splits the atomic points dataset into two groups. The first left branch group with points whose x-axis values are less than or equal to the atomic root point $x_{C[1]}$. While the other right branch group with points whose x-axis value is greater than $x_{C[1]}$. The split at the atomic root point is visualized in Figure 4. The red line is the splitting line according to the x-axis.

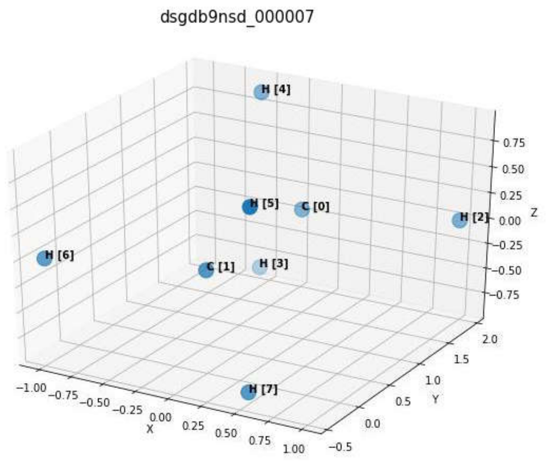

Figure 2. Eight atoms of the molecule *dsgdb9nsd_000007* in the XYZ cartesian coordinate system.

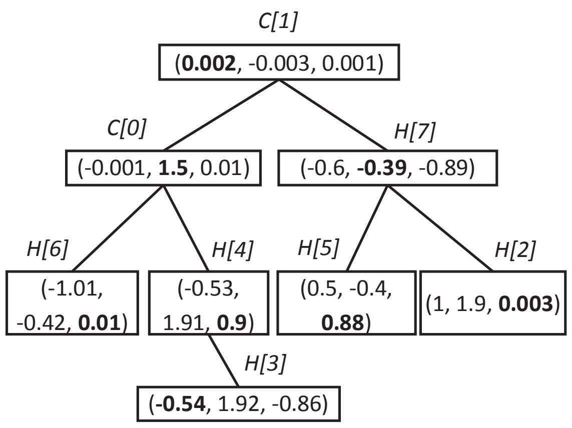

Figure 3. Visualization of the K-D tree structure constructed from 8 atomic points. Each square contains the XYZ cartesian coordinate with its *atom[index]*.

Then, to continue building the K-D tree for the molecule *dsgdb9nsd_000007*, recursively build the K-D tree in the right and the left half-space in Figure 4 by splitting at the atomic point hydrogen *(i=7)* of the right half-space and hydrogen (*i=4)* of the left half-space. The reason is that both mentioned atomic points are the median point according to the y-axis, and splitting remained atomic data point horizontally through it. Continuing partition recursively to completion will result in the entirely constructed K-D tree, as illustrated in Figure 5. The blue line is the splitting line according to the y-axis, while the green line is the splitting line according to the z-axis.

From the entirely constructed K-D tree, the K-D tree query has been implemented to get the array list of ordered atoms index relying on the distance from the selected atom point. Since the ordered atoms list of the molecule *dsgdb9nsd_000007* is: *['H', 'H', 'H', 'H', 'H', 'H', 'C', 'C']*, we start with the atom H at *i=2*. For each atom, the K-D tree query returns the nearest neighbor according to the distance. Based on the entirely constructed K-D tree structure in Figure 5, the atom C at *i=1* (C[1]) as the root is taken as an example. Atom C[1] becomes the query point in the K-D tree structure. The C[1] query point continually traverses all nodes inside its branch then creates its sphere where the radius is the distance from C[1] to the current nearest node. Next, we check whether the sphere crosses any coordinate axis, backtrack to the intersected branch, and measure the distance to find the more current nearest node. We repeatedly measure, create the new sphere, and check the intersected area until the nearest node is found. In this case of C[1], the atom hydrogen with *i=6* (H[6]) becomes the nearest neighbor, which is shown in Figure 6. The distance from C[1] to H[6] is 1.09495347.

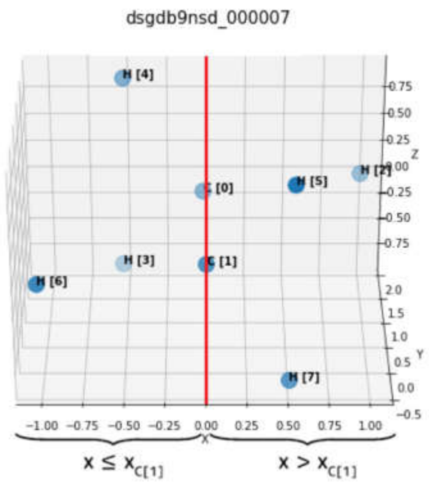

Figure 4. Visualization of splitting the atomic points into two groups according to the x-axis at atom C *(i=1)*.

After finding out H[6] as the nearest node of C[1], if there is no bond yet between those two atoms and the connection between them certainly exists once, the algorithm continues to compare the calculated distance between them to the predicted bonding distance. In addition, the number of nearest nodes taken into consideration is based on the valence of the element. If the valence value is larger than the number of atoms in a specific molecule, the maximum number of atoms is used. For example, the valence of C is 4, and then the four nearest nodes are taken to measure the distance.

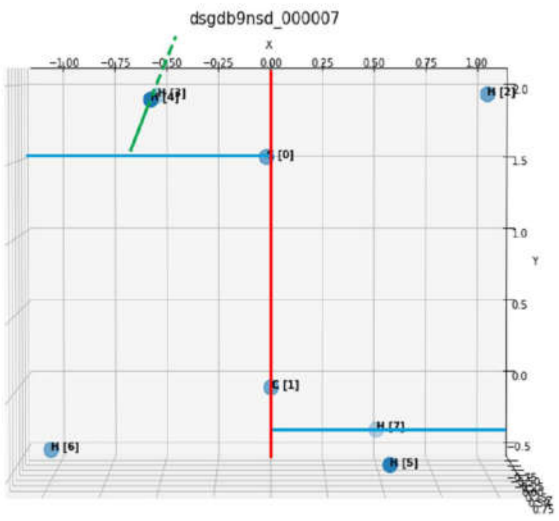

Figure 5. An entirely constructed K-D tree applies to the molecule *dsgdb9nsd_000007*.

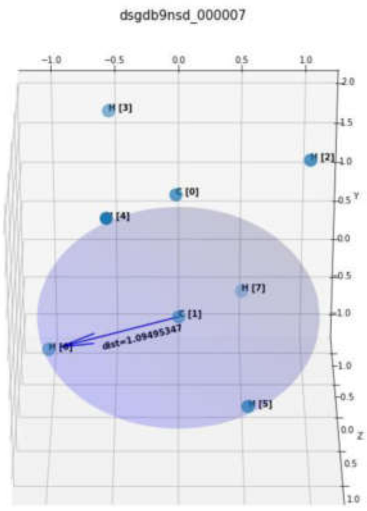

Figure 6. The nearest neighbor of atom C *(i=1)* is the atom H *(i=6)* with the *dist=1.09495347*.

The predicted bonding distance between them is measured as the sum of the bond length of each atom. In this case, only atoms with the measured distance in the 20% expected distance or closer are kept. Then, we check whether both atoms have remaining valence or not and continue decreasing the remaining valence and creating a new bond. If any of them has zero remaining valences, we mark both atoms as leaves. The function continues running until all nodes are marked as leaves. In the end, the molecule *dsgdb9nsd_000007* is constructed as in Figure 7. Based on the fully bonding reconstructed structure of *dsgdb9nsd_000007*, it is relatively easy to figure out that the chemical formula is Ethane (C_2H_6), all connected by single bonds.

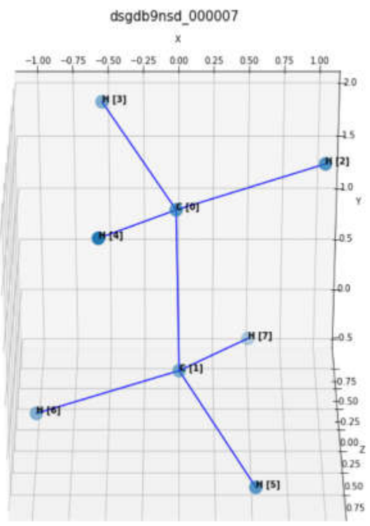

Figure 7. Fully bonding reconstructed *dsgdb9nsd_000007* molecule.

2.2.2. n-bond connect greedy reconstruction

After successfully reconstructing the single bonding, the next step is rebuilding n-bond connections. Even though there is a calculated table of bond length for the double bond and triple bond, the accuracy and the consistency of the bonding connection are low. For example, the C atom has various variations in CC bond lengths that can be reasonably explained on the basis of hybridization being the primary factor. When atoms with lone pairs are involved, it appears necessary to introduce electron delocalization effects.[21] As a consequence, the bond length and bond angles do not provide a reliable measure of carbon hybridization. Any atom whose valence is greater than 1 can potentially share two or three pairs of electrons with another atom. The n-bond connect reconstruction is applied for any molecule having any remaining available valence, which is not yet connected to other atoms. The molecule *dsgdb9nsd_000005* is a typical example for this case because after applying the single bond connect reconstruction, its total remaining available valence is 4. The structure data of molecule *dsgdb9nsd_000005* are listed in Table 2. Both atom C and atom N have two remaining unconnected bonds. Especially, atom C is marked as a leaf due to its connection to atom H, which has zero valence.

The idea of solving the n-bond connection is to link the atoms by electron pair bonds until each atom has a full octet based on the Lewis structure for compounds. While there are remaining atoms marked as leaves with available valence, the algorithm will add as many bonds as possible between atoms having any available valence. Every atom is taken into consideration one by one. Until any of them has remaining valences, the algorithm will mark both atoms as leaves. The greedy algorithm is also applied to make a locally optimal choice at each stage. If any atoms still have a remaining available valence, the algorithm will check each key according to these bonding keys. Then, the bond will be added to as many as possible between a pair of atoms that have an available valence. With the greedy algorithm, it is possible to entirely reconstruct the double bond and triple bond of molecules.

Table 2. The XYZ cartesian coordinate of the molecule *dsgdb9nsd_000005.*

index	atom	x	y	z
0	C	-0.0133	1.1324	0.0082
1	N	0.0023	-0.0191	0.0019
2	H	-0.0278	2.1989	0.0141

To better understand how the implemented algorithm works, there is a visualization of the bonding schema of the molecule *dsgdb9nsd_000005*. As can be seen from Figure 8, the bonding between atom C at *i=0* (C[0]) and atom H at *i=2* (H[2]) is colored black, which means the single bond. Since the H[2] is marked as a leaf, so does the C[0]. Next, the algorithm solves for the bonding between C[0] and atom nitrogen at *i=1* (N[1]). After running the n-bond connect greedy reconstruction, the triple bond between C[0] and N[1] is constructed as the red line. All atoms in the molecule have no available valence left. Hence, the double bond and triple bond reconstruction algorithm is assumptively successful.

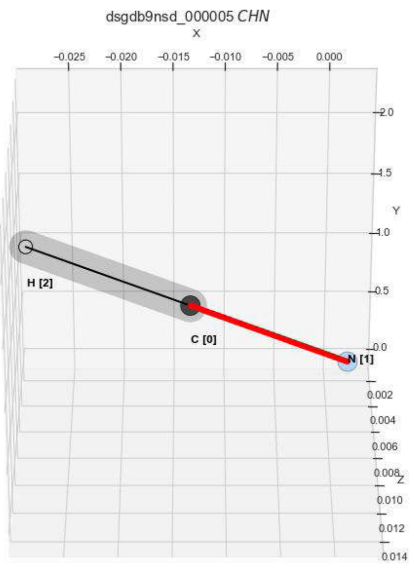

Figure 8. Fully bonding reconstructed *dsgdb9nsd_000005* molecule. The black line presents a single bond, and the red line presents the triple bond.

2.2.3. Ionized radical search

After successful n-bond reconstruction, there are still many molecules without the completed bonding structure due to the ionization. The possible ionized groups which can be formed from H, C, N, F, and O in the dataset are Carboxyle (COO^-) and Ammoniumyl (NH_3^+). The idea of searching ionic bonds is initially to look for covalent bonds with remaining valence on some atoms where these covalent bonds are processed n-bond connection. For example, to find the ionic group NH_3^+, it is necessary to search for the disconnected NH_3. However, for the ionic group COO^-, we need to find the CO group with one available bond connected to an O atom.

To better understand the radical ionic search algorithm, the molecule *dsgdb9nsd_000271* is taken into consideration because it has both ionic group COO^- and NH_3^+ in its structure. After pre-processing, the molecule *dsgdb9nsd_000271* is identified as Alanine with the chemical formula $C_3H_7NO_2$. After processing with the ionic radial search algorithm, the structural result of the molecule *dsgdb9nsd_000271* is visualized in Figure 9.

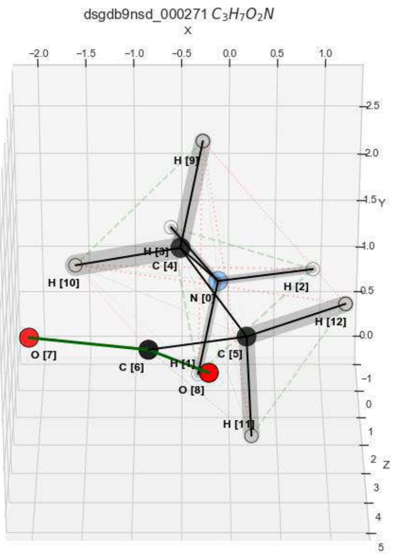

Figure 9. Fully bonding reconstructed *dsgdb9nsd_000271* *($C_3H_7NO_2$)* molecule. The black line presents the single bond, the green line presents the double bond, and the red line presents the triple bond.

2.2.4. Ring search

Besides reconstructing the bonding schema, the ring of atoms inside a particular molecule should be considered. The reason for identifying the chemical ring is to easily distinguish the correct chemical formula of a specific molecule. For instance, the molecule *dsgdb9nsd_000017* is one of the remarkable molecules containing a ring inside itself. The reason this molecule is more specific than the other is that the chemical formula formed from its atoms can be three different compounds.

Table 3. The XYZ cartesian coordinate of the molecule *dsgdb9nsd_000017.*

index	atom	x	y	z
0	C	0.0153	1.4176	0.009
1	C	1.2648	0.6492	-0.0066
2	O	-0.0002	-0.0077	0.002
3	H	-0.3176	1.8859	0.9348
4	H	-0.3353	1.8958	-0.9039
5	H	1.8324	0.5626	-0.9319
6	H	1.8501	0.5527	0.9068

The approach of the algorithm to identify the ring of a particular molecule and its order is to apply the network graph to search for a cycle graph. The minimum cycle basic algorithm supports this approach since searching for the chemical ring is equivalent to finding the minimal cycle basic in a graph where the graph is the bonding structure. It is a cycle basic for which the total weight, in other words, the length for an unweighted graph, of all the cycles is minimum. The graph is split into connected subgraphs. The idea behind the minimum cycle basic algorithm is to use an all-pairs shortest paths (APSP) algorithm as a subroutine. Then, Dijkstra's algorithm is used for APSP computation. In other words, the graph of the bonding structure will be analyzed to find the shortest ring in a molecule.

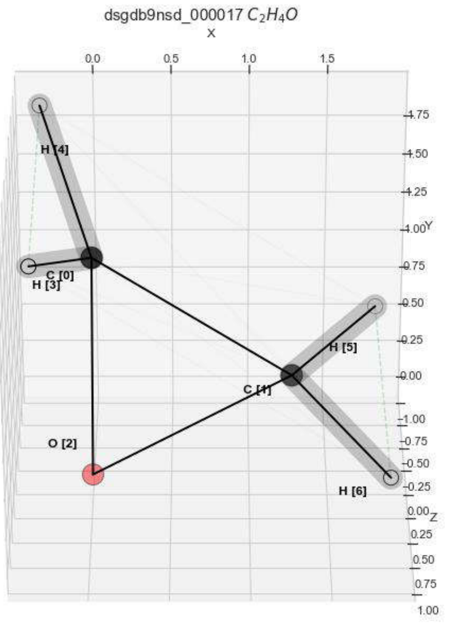

Figure 10. Fully bonding reconstructed *dsgdb9nsd_000017* molecule (C_2H_4O). The ring is between C, O, and C.

3. RESULTS AND DISCUSSION

3.1. Results

After we apply the KNN-KD tree algorithm in the construction of molecular structures, we get 45772 unique molecules built from the test set and 85003 unique molecules built from the train set. After constructing the bonding type, there are 17 bonding types.

The number of built molecules is equal to the initial number of molecules in the dataset, which means it is likely that there is no failure in the reconstruction algorithm. However, there is a high possibility that some molecules are not correctly handled according to the order of atoms or the bonding types. That is why there should be a proper evaluation to determine the success rate of the bonds reconstruction algorithm.

To evaluate the success rate of our KNN-KD tree algorithm, we propose to compare the output results to the result computed by OpenBabel, which is a chemical toolbox designed to search, analyze, convert, or store data from molecule modeling, chemistry, solid-state materials, biochemistry, or related areas.[22] Besides that, we conduct another baseline to test the accuracy of the KNN-KD tree algorithm. The second evaluation is to calculate the bond type consistency based on the distribution of bond length.

3.1.1. Evaluate by pairwise comparison versus OpenBabel

The pairwise comparison is applied to evaluate the calculated bonding type related to the bonding distance between two atoms and whether they are significantly different from one another. A pairwise-comparison trial included a pair of scalar coupling with its bonding type and the bonding distance between atoms. We derive results from the OpenBabel toolbox with the CHAMPS dataset and compare them with results computed by our KNN-KD tree algorithm. Any pair having a different bond type or a significant difference in the distance value between two atoms is set as an error, which is considered to be larger than 0.01.

Table 4 shows that there are 31927 scalar coupling observations of 2064 unique molecules marked as the error, which accounts for 2.4% of total processed molecules in the train set. At the same time, there are 17692 scalar coupling observations of 1150 unique molecules flagged as the error, which occupies around 2.5% of total processed molecules in the test set.

Table 4. The comparison table between our method and OpenBabel.

Dataset	Test set	Train set
Unique molecule	45772	85003
Inconsistent unique molecule	1150	2064
Unconsistency percentage	2.5%	2.4%

3.1.2. Evaluate by bond type consistency distribution

Each bonding pair is grouped by its different bonding valence for both the train and test datasets. So, it is more understandable to analyze the distribution of bonding types over the distance between atoms. The computation of bond type consistency based on the bond length is applied. Each bonding pair is grouped by its different bonding valence for both the train and test datasets. So, it is more understandable to analyze the distribution of bonding types over the distance between atoms.

As can be seen from Figure 14, the train and test set match well for the relative distribution of bond length. The train and test should be the same for the prediction of the scalar coupling constant. The number of bonds also builds the distributions, which peak at a different bond distance. This is consistent with the expected behavior that the more bonds, the further distance between two atoms.

Table 5. The successful rate of bond reconstruction according to coupling types

Dataset	Coupling type	Running time	Successful rate
Train set	1JHC	111s	100.00%
	1JHN	141s	100.00%
	2JHC	241s	99.98%
	2JHH	336s	100.00%
	2JHN	379s	99.88%
	3JHC	491s	99.97%
	3JHH	600s	99.96%
	3JHN	655s	99.94%
Test set	1JHC	702s	100.00%
	1JHN	717s	100.00%
	2JHC	768s	99.99%
	2JHH	817s	100.00%
	2JHN	839s	99.90%
	3JHC	896s	99.98%
	3JHH	951s	99.96%
	3JHN	980s	99.96%

The distribution of the 1.0 CC, 2.0 CC, and 3.0 CC is a perfect example of very well separated distributions. Adding the COO^- handling improved a lot of things by separating the previous 1.0 CO bimodal distribution into two well-defined peaks, one for 1.5 CO and the second for 1.0 CO. Consequently, some other bimodal distributions like 1.0 CN, which decreased in 6.5% of bonds, can be expected to be resolved the same way if needed. Based on the calculated distribution, the running time of the implemented algorithm, together with the success rate, can be derived in Table 5.

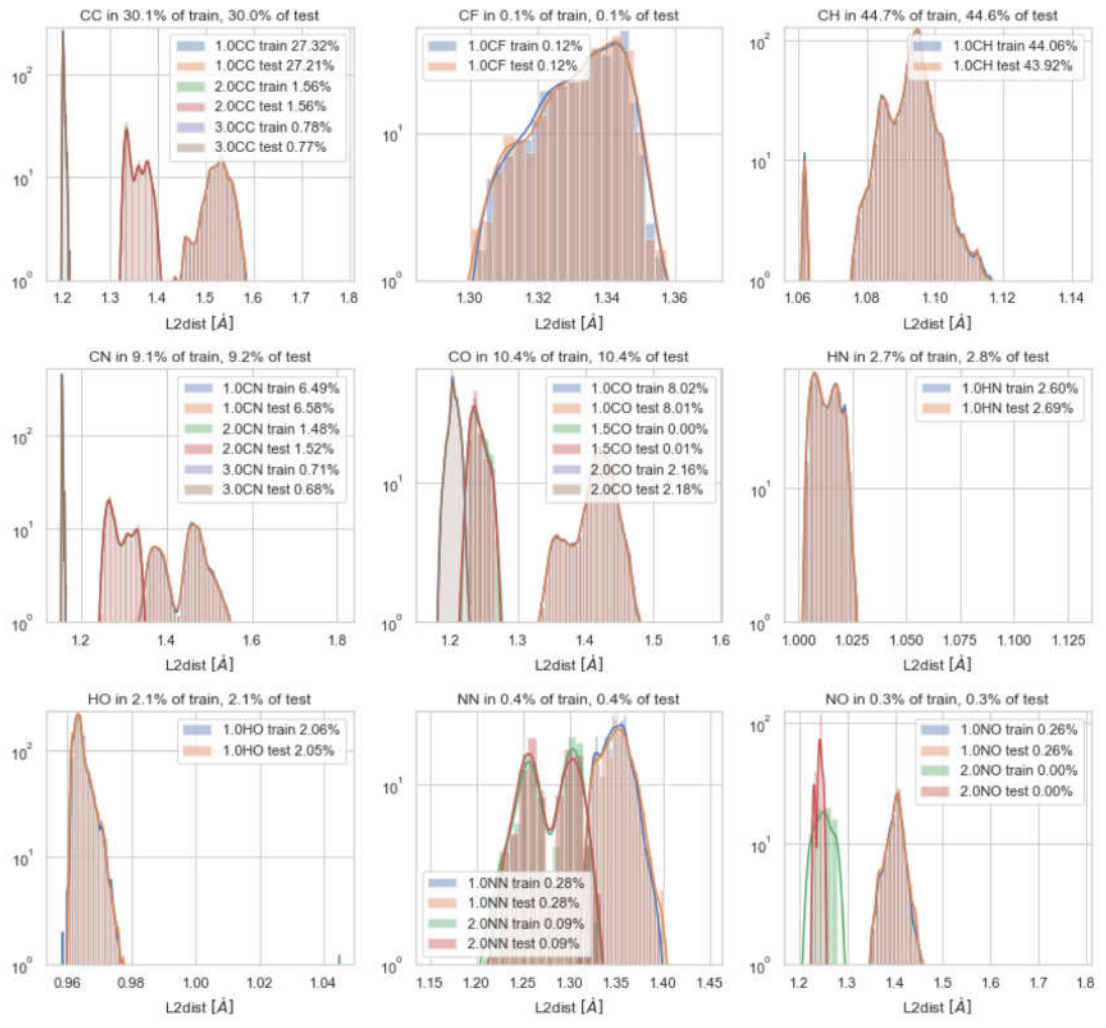

Figure 11. The diagram of bond length by atom pair and number of bonds. The line illustrates the distribution of each bond type in the train/test set.

3.2. Discussion

Our results can be improved by applying other systems. The initial algorithm calculates the distance between atoms based on the XYZ cartesian system. However, according to some research, this representation is not stable. Each coupling pair is located at a different point in space, and two similar coupling sets would have significantly different XYZ positions. So, instead of using coordinates, another system is considered. In this system, initially, each pair of atoms is taken as two first core atoms. The distance of the center between each pair needs to be calculated. Then, all n-nearest atoms to the center, which exclude the first two atoms, are required. Any two closest atoms become the third and the fourth core atoms. Finally, the distances from four core atoms to the rest of the atoms and to the core atoms are calculated as well. By using this representation, each atom's position can be described by four distances from the core atoms. This representation is not only stable for rotation and transition but also suitable for pattern-matching. So, by taking a sequence of atoms together with describing each by four distances and atom type and looking up for the same pattern, we can find similar configurations and detect the scalar coupling constant.

4. CONCLUSION

This research of analyzing and visualizing the molecular properties based on the KNN-KD tree algorithm has confirmed that our method can successfully construct the structure of molecules with a comparable result to rule-based methods. The findings also revealed that taking some additional datasets into account can improve the success rate of constructing the molecular structures, such as dipole interactions, magnetic shielding, and potential energy, Mulliken charges.[23–25] With the benchmark studies, the advantages and disadvantages of some data structures, which are also used for distance calculation, are presented. With our KNN-KD tree algorithm in the construction of molecular structures, utilizing models to predict the scalar coupling constants has become much more straightforward and correct. These facts motivate us to conduct and investigate the relationship between atoms in a particular molecule in the future further.

In conclusion, this research makes the following contributions: 1) proposing a geometric-based approximated machine learning model, namely the KNN-KD tree in the construction of molecular structures only with XYZ coordinates of atoms for training the model. Unlike other data structures for distance calculation, our method reduces the pre-processing time, single query time and computational resources in various essential chemistry fields such as biomedical engineering, drug discovery, and vaccine exploration. 2) visualizing the molecular structure based on the bonding schema, which was built by our KNN-KD tree algorithm, to give a better understanding and representational figures. 3) leveraging the force field method in molecular modeling because it can be extended to estimate the forces and potential energy of a system of atoms.

Our future works concern a more in-depth analysis of particular mechanisms and new proposals to try different methods. Although the results of the proposed algorithm are reasonable, there is still room for improvement. There are some features and some additional datasets that are not accounted for that likely have a significant effect on each different bonding type. The algorithm may take dipole interactions, magnetic shielding, potential energy, and Mulliken charges into account to improve the accuracy.

REFERENCES

1. Predicting molecular properties, <https://kaggle.com/c/champs-scalar-coupling>, accessed 21/01/2022.

2. X. Yang, Y. Wang, R. Byrne, G. Schneider, S. Yang. Concepts of artificial intelligence for computer-assisted drug discovery, *Chemical Reviews,* **2019**, *119*(18), 10520-10594.

3. K. Hansen, F. Biegler, R. Ramakrishnan, W. Pronobis, O. A. von Lilienfeld, K. - R. Müller, A. Tkatchenko. Machine learning predictions of molecular properties: accurate many-body potentials and nonlocality in chemical space, *Journal of Physical Chemistry Letters*, **2015**, *6*(12), 2326-2331.

4. K. Gubaev, E. V. Podryabinkin, A. V. Shapeev. Machine learning of molecular properties: locality and active learning, *The Journal of Chemical Physics,* **2018**, *148*(24), 241727.

5. boschresearch/BCAI_kaggle_CHAMPS, <https://github.com/boschresearch/BCAI_kaggle_CHAMPS>, accessed 21/01/2022.

6. A. Yoshimori. Prediction of molecular properties using molecular topographic map, *Molecules,* **2021**, *26*(15), 4475.

7. J. Li, X. Jiang. Mol-BERT: An effective molecular representation with BERT for molecular property prediction, *Wireless Communications and Mobile Computing,* **2021**, *2021*, e7181815.

8. X. - Y. Yu, R. Fu, P. - Y. Luo, Y. Hong, Y. - H. Huang. Construction and prediction of antimicrobial peptide predicition model based on BERT, <https://jasonyanglu.github.io/files/lecture_otes/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0_2020/Project/Construction%20and%20Prediction%20of%20Antimicrobial%20Peptide.pdf>, accessed 13/02/2022.

9. J. C. Bezdek, S. K. Chuah, D. Leep. Generalized K-nearest neighbor rules, *Fuzzy Sets and Systems* **1986**, *18*(3), 237-256.

10. J. L. Bentley. Multidimensional binary search trees used for associative searching, *Communication of the ACM,* **1975**, *18*(9), 509-517.

11. J. S. Beis, D. G. Lowe. *Shape indexing using approximate nearest-neighbour search in high-dimensional spaces*, In Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1997, 1000-1006.

12. M. Otair. Approximate K-nearest neighbour based spatial clustering using K-D tree, *International Journal of Database Management Systems*, **2013**, *5*(1), 97-108.

13. Nguyen Truong Thanh Hung. *Analyzing and visualizing chemical molecular properties using K-Dimension tree algorithm*, BA thesis, 2020.

14. M. Muja, D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **2014**, *36*(11), 2227-2240.

15. G. S. Iwerks, H. Samet, K. Smith. - *Continuous K-nearest neighbor queries for continuously moving points with updates*, Proceedings 2003 VLDB Conference, Freytag, San Francisco, 2003, 512-523.

16. A. J. Broder. Strategies for efficient incremental nearest neighbor search, *Pattern Recognition,* **1990**, *23*(1), 171-178.

17. W. Hou, D. Li, C. Xu, H. Zhang, T. Li. *An advanced K-nearest neighbor classification algorithm based on KD-tree*, 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), 2018, 902-905.

18. M. Tafazzoli, M. Ghiasi. New karplus equations for 2JHH, 3JHH, 2JCH, 3JCH, 3JCOCH, 3JCSCH, and 3JCCCH in some aldohexopyranoside derivatives as determined using nmr spectroscopy and density functional theory calculations, *Carbohydrate Research,* **2007**, *342*(14), 2086-2096.

19. R. D. Shannon. Revised effective ionic radii and systematic studies of interatomic distances in halides and chalcogenides, *Acta Crystallographica Section A*, **1976**, *32*(5), 751-767.

20. Z. Li, C. Bommier, Z. S. Chong, Z. Jian, T. W. Surta, X. Wang, Z. Xing, J. C. Neuefeind, W. F. Stickle, M. Dolgos, P. A. Greaney, X. Ji. Mechanism of Na-Ion storage in hard carbon anodes revealed by heteroatom doping, *Advanced Energy Materials,* **2017**, *7*(18), 1602894.

21. D. R. Lide. A survey of carbon-carbon bond lengths, *Tetrahedron,* **1962**, *17*(3), 125-134.

22. N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch, G. R. Hutchison. Open babel: An open chemical toolbox, *Journal of Cheminformatics,* **2011**, *3*(1), 33.

23. M. R. Manaa, H. A. Ichord, D. W. Sprehn. Predicted molecular structure of novel C48B12, *Chemical Physics Letters,* **2003**, *378*(3), 449-455.

24. Z. Rinkevicius, J. Vaara, L. Telyatnyk, O. Vahtras. Calculations of nuclear magnetic shielding in paramagnetic molecules, *Journal of Chemical Physics,* **2003**, *118*(6), 2550-2561.

25. V. P. Spiridonov, N. Vogt, J. Vogt. Determination of molecular structure in terms of potential energy functions from gas-phase electron diffraction supplemented by other experimental and computational data, *Structural Chemistry,* **2001**, *12*(5), 349-376.