

# Experimental Studies of the Universal Chemical Key (UCK) Algorithm on the NCI Database of Chemical Compounds

Robert Grossman<sup>†</sup>

*Laboratory for Advanced Computing  
University of Illinois at Chicago  
Chicago, IL 60607  
grossman@uic.edu*

Pavan Kasturi

*Laboratory for Advanced Computing  
University of Illinois at Chicago  
Chicago, IL 60607  
pavan@lac.uic.edu*

Donald Hamelberg

*Laboratory for Advanced Computing  
University of Illinois at Chicago  
Chicago, IL 60607  
don@lac.uic.edu*

Bing Liu

*Department of Computer Science  
University of Illinois at Chicago  
Chicago, IL 60607  
liub@cs.uic.edu*

## Abstract

*We have developed an algorithm called the Universal Chemical Key (UCK) algorithm that constructs a unique key for a molecular structure. The molecular structures are represented as undirected labeled graphs with the atoms representing the vertices of the graph and the bonds representing the edges. The algorithm was tested on 236,917 compounds obtained from the National Cancer Institute (NCI) database of chemical compounds. In this paper we present the algorithm, some examples and the experimental results on the NCI database. On the NCI database, the UCK algorithm provided distinct unique keys for chemicals with different molecular structures.*

## 1. Introduction

Chemical compounds usually have several common names. Although unique identifiers attached to chemical compounds would be useful for a variety of purposes, there is no consensus about how to do this. Currently most nomenclatures for chemical compounds either do not provide unique keys or the unique keys provided are based upon convention, such as when the compound was entered into a database. For this reason, determining whether a compound was entered into a database twice or comparing compounds across databases is difficult.

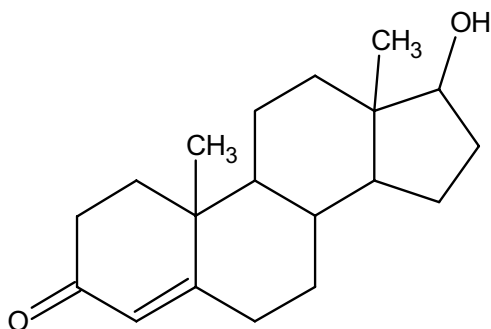
An illustration of the structural formula of such a compound, Testosterone, is depicted as an example in

Figure 1. This molecule as entered in the National Cancer Institute (NCI) database of chemical compounds has 54 names associated with it and a unique id of 9700, which is also different from its Chemical Abstract Services (CAS) id of 58-22-0. Because of examples like this, it is very important to construct a unique key that is derived from the structural features of the compound. Using such a key, properties of a chemical contained in a database in one location could be combined with properties of the same chemical compound contained in a database in another location automatically. With the increasing use of distributed infrastructures for computing, such as data grids and web service-based platforms, having universal chemical keys that can be used to combine distributed data about chemical compounds is of growing importance. Indeed, we have used the UCK algorithm described here to build distributed data web applications for docking chemical compounds in proteins from the Protein Data Bank (PDB).

Our algorithm for computing what we call a Universal Chemical Key or UCK is based upon abstracting the chemical compound as a labeled graph, with atoms represented by nodes and bonds represented by edges. The nodes are labeled with the symbols corresponding to the atoms they represent. Note that two labeled graphs representing molecular structures are the same or isomorphic if they are labeled using the same labels and can be mapped onto each other such that the labels of nodes or atoms and edges or bonds are conserved.

We introduce an algorithm, which given a labeled graph representing a chemical compound, produces a long string, which is the UCK. This string has the properties:

i) Chemical compounds associated with the same labeled graph are identical and produce the same UCK. ii) The UCKs of different labeled graphs are different in practice according to our experiments. Since the problem of distinguishing labeled graphs in general is NP-hard, it is not reasonable to expect a fast algorithm to do this 100% of the time. On the other hand, we show that in practice, on large collections of chemical compounds such as the NCI database, our UCK algorithm does have this property. Since the UCK strings can be quite long, we associate a shorter string using a standard hashing algorithm called MD5 [1]. Although the MD5 hash is not guaranteed to be unique, in practice it almost always is unique.



**Figure 1.** Structural formula of Testosterone, C<sub>19</sub>H<sub>28</sub>O<sub>2</sub>. Testosterone has the NSC id of 9700 and the CAS id 58-22-0. Some of the other names Testosterone goes by are 17-hydroxyandrost-4-en-3-one, Androlin, Cristerona T, and Homosteron.

In this paper, we present some background of the problem for finding unique keys or nomenclatures for chemical compounds in the Related Work section. Our algorithm is presented in the Computational Methods and Algorithm section. We tested the algorithm on the National Cancer Institute (NCI) database of chemical compounds, and the results are detailed in the Results section with some relevant examples. In closing we give a brief summary of this study and discuss some future work.

## 2. Related work

Numbering and ordering of atoms and groups of atoms of molecular structures have always been done by organic chemists, which subsequently led to several different systems of naming compounds. The International Union of Pure and Applied Chemistry (IUPAC) nomenclature rules [2] are the most widely used. However, these rules only work effectively for very small molecules and generally are inconsistent, hard to understand, and easily cause mistakes [3, 4, 5]. The idea

of using graph theory in solving the above problem was put forward some twenty years ago [6, 7], but never really caught the attention of chemists.

Recently, the International Union of Pure and Applied Chemistry (IUPAC) established a project to develop unique keys for chemical compounds. This is an ongoing project, which was announced [8] a couple of years ago. The approach that the scientists are using is in part based on graph theory. However, the details are yet to be published. The documentation and algorithm will be published at the end of the project. Their aim is to generate a unique key from a graphical input of structural information of known and as yet unknown compounds. The keys, known as IUPAC Chemical Identifiers are alphanumeric text strings that can lead back to the structure of the compound and that can be digitized to be used in printed and electronic information.

In calculating a unique identifier for a chemical compound, one can represent the molecule as a graph as discussed in the Introduction. Therefore, two molecules are the same if the graph representations are isomorphic. Several algorithms [9, 10, 11] have been developed to check whether two graphs are isomorphic. However, direct comparison of molecular structures will not be adequate for this study because the aim is to develop an algorithm that is capable of generating unique ids for chemical compounds whether they are presently available or not. The chemical id problem can be solved if one can uniquely order the nodes in a labeled graph irrespective of their original order. These types of algorithms are in fact very useful in the graph isomorphism problem. Tinhofer and coworkers [12] developed an algorithm to generate canonical numbers for a labeled graph. They do this by considering all adjacent matrices of the graph that belong to the isomorphism class. Each matrix is read row by row as a binary number and the matrix with the smallest number is selected. The graph with this numbering is considered to be canonically numbered. Also, Brendan D. McKay [13, 14] has developed an algorithm to generate a canonical labeling map of a labeled graph. This algorithm has been implemented in the program called "nauty."

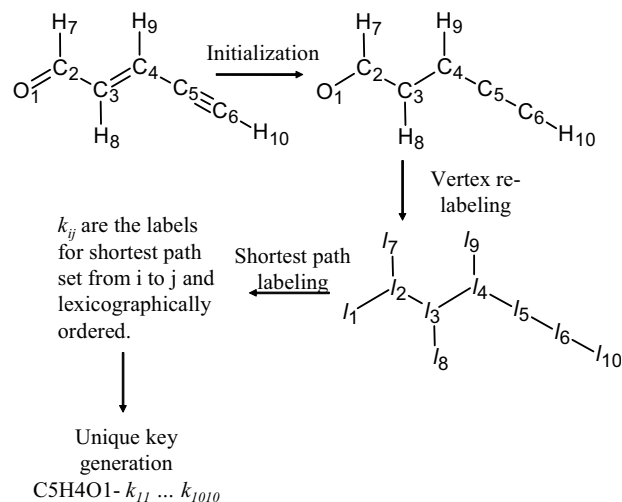
## 3. Computational methods and algorithms

### 3.1 Overview

We begin with an overview of the algorithm. The first step in generating the UCK is to represent different types of covalent bonds, i.e., single, double or triple by just a single bond between atoms of the molecule, so that we can focus only on the connectivity and also because there is no unique way to determine double and triple bonds. We also disregard the overall charge on the molecule. If desired, variants of the algorithm below could incorporate this information as additional labels.

The molecule is now considered as an undirected labeled graph  $G = (V, E)$  consisting of a finite set of vertices  $V$  and a finite set of edges  $E$ . The set of atoms in the molecule is the vertex set,  $V$  and the set of covalent bonds between the atoms is the edge set,  $E$ . Each edge has a unity weight. At this stage, the labels in the graph  $G$  represent atoms and do not reflect any of the local structure around the atom.

The next step in the UCK algorithm is to replace the labels with new labels that capture some of the local connectivity and chemical environment around each atom. Atoms with similar connectivity will end up having similar labels. In the next step of the UCK algorithm, the lengths of the shortest paths between each pair of vertices of the vertex set  $V$  of the graph  $G$  are generated. The path labels are produced by concatenating the source label, the path length, and the destination label. At every stage of labeling we follow a rule based lexicographical ordering so that the whole procedure is invariant to the changes in ordering of the vertex set  $V$ . A lexicographical ordering of the labels of all pair of shortest path sets is done. The labels are concatenated to form a string and prefixed by the molecular formula of the molecule. The string thus obtained uniquely represents the molecule.



**Figure 2.** An example showing the procedure to generate a unique string for a given molecule.

Here is part of an example. A more detailed description of the algorithm follows. A schematic representation of a molecule that may or may not exist is shown in Figure 2 with single, double and triple bonds and shows an overview of the algorithm. We represent the double and triple bonds as single bonds, so that it can be represented as a graph  $G = (V, E)$  where the vertex set  $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$  and the edge set  $E = \{(1,2), (2,3), (3,4), (4,5), (5,6), (2,7), (3,8), (4,9), (6,10)\}$ . The

list  $\langle O, C, C, C, C, C, H, H, H, H \rangle$  is the label list for elements corresponding to  $V$ . Observe that though the connectivity of the second vertex is different from that of the third to the sixth they all have the same labels.

## 3.2 Algorithm

### 3.2.1 Initial setup

The first step is to represent different types of covalent bonds as single bonds. Next we consider the molecule as an undirected graph  $G = (V, E)$  consisting of a finite set of vertices  $V$  and a set of edges  $E$ . The set of atoms in the molecule is the vertex set,  $V$  and the set of covalent bonds between the atoms is the edge set,  $E$ . The weight function is defined as  $w: E \rightarrow \{1\}$ .

### 3.2.2 Labeling of the vertices

For a depth  $d$ , we define  $\lambda^{(d)}$  inductively from  $\lambda^{(d-1)}$ .  $\lambda^{(d)}$  re-labels each node in  $G$ .

Step 0:

Define  $\lambda^{(0)}(b) = \text{label}(b)$ , where  $b$  is a node in Graph  $G$  and  $\lambda^{(d)}$  is the label map.

Step 1:

Fix node 'a' and let  $b_1, \dots, b_k$  be its children.

Step 2:

Compute  $\lambda^{(d-1)}(b_1), \dots, \lambda^{(d-1)}(b_k)$

Step 3:

Remove all occurrences of  $a$  from  $\lambda^{(d-1)}(b_1), \dots, \lambda^{(d-1)}(b_k)$ , lexicographically order them to produce the string  $\lambda^{(d-1)}(b_{i1}) \dots \lambda^{(d-1)}(b_{ik})$ .

Step 4:

Define  $\lambda^{(d)}(a) = \text{label}(a) \lambda^{(d-1)}(b_{i1}) \dots \lambda^{(d-1)}(b_{ik})$

### 3.2.3 Labeling of shortest paths

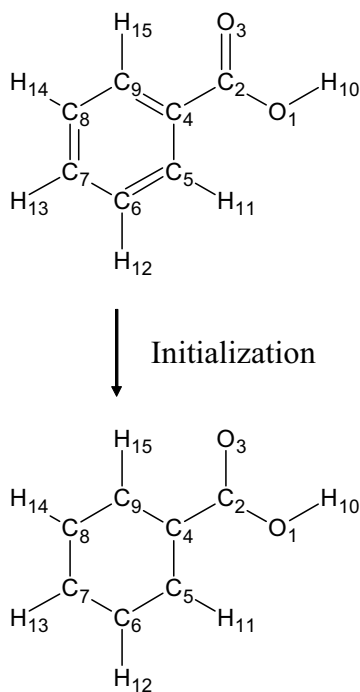
Re-label  $G$  with labels  $\lambda^{(2)}$  (we use  $d = 2$  in this work). Call this  $G'$ . Fix  $a \in V$  of  $G'$  for each  $b \in V$  of  $G'$ . Define  $\mu(b) = \langle \text{label}(a) \text{ length of shortest path from } a \text{ to } b \rangle$ , where  $n = \text{length of shortest path from } a \text{ to } b$ .

### 3.2.4 Generate unique key $\mu(G)$ for graph $G$

$\mu(G) = \text{Chemical formula-Lexicographically Ordered string } \langle \mu(b) \mid a \in V, b \in V \text{ of } G' \rangle$   $\mu(G)$  is the unique key.

### 3.3 Example

Let us consider benzoic acid ( $C_7H_6O_2$ ).



**Figure 3.** Structure of benzoic acid (NCI number = 149).

The new label for the vertices is shown in Table 1 below.

**Table 1.** Showing new labels generated.

Vertex Number	New Label $\lambda^{(2)}$
1	OCCOH
2	CCCCOOH
3	OCCO
4	CCCHCCHCOO
5	CCCCCCHH
6	CCCHCCHH
7	CCCHCCHH
8	CCCHCCHH
9	CCCCCCHH
10	HOC
11	HCCC
12	HCCC
13	HCCC
14	HCCC
15	HCCC

The unique string  $\mu(G)$  is shown below:

C7H6O2-CCCCCCHH0CCCCCCHHCCCCCCHH0CCCCCCHHCCC  
 CCCCCHH1CCCCHCCHCOOCCCCCCHH1CCCCHCCHCOOCCCCCCHH  
 1CCCCHCCHHCCCCCCHH1CCCCHCCHHCCCCCCHH1HCCCCCCH  
 CCHH1HCCCCCCHH2CCCCCCHHCCCCCCHH2CCCCCCHH  
 CCCCCCHH2CCCCOCHCCCCCCHH2CCCCOCHCCCCCCHH2CC  
 CHCCHHCCCCCCHH2CCCCCHCCHHCCCCCCHH2HCCCCCCHH  
 H2HCCCCCCHH3CCCCHCCHHCCCCCCHH3CCCCHCCHHCCC  
 CCHH3HCCCCCCHH3HCCCCCCHH3HCCCCCCHH3HCCCCCCH  
 H3HCCCCCCHH3OCCOCCCCCCHH3OCCOCCCCCCHH3OC  
 COHCCCCCCHH3OCCOCHCCCCCCHH4HCCCCCCHH4HCCC  
 CCCCCCHH4HOCCCCCCHH4HOCCCCCCHH4OCCCCOCHCCCC  
 OOH1CCCCHCCHCOOCCCCOCH1OCCOCCCCOCH1OCCOCHCCC  
 OOH2CCCCCCHHCCCCOCH2CCCCCCHHCCCCOCH2HOCCCC  
 OOH3CCCCHCCHHCCCCOCH3CCCCHCCHHCCCCOCH3HCCCCC  
 COOH3HCCCCCCHH4CCCCCHCCHHCCCCOCH4HCCCCCCHCO  
 H4HCCCCCCHCOOH5HCCCCCCHCCHCOO0CCCCHCCHCOO  
 CCHCO01CCCCCCHHCCCCCHCCHCO01CCCCCCHHCCCCCHCCHCO  
 01CCCCOCHCCCCCHCCHCO02CCCCHCCHHCCCCCHCCHCO02CCC  
 CHHCCCCCHCCHCO02HCCCCCCHCCHCO02HCCCCCCHCCHCO  
 02OCCOCCCCCHCCHCO02OCCOCCCCCHCCHCO03CCCCHCCHHCC  
 CHCCHCO03HCCCCCCHCCHCO03HCCCCCCHCCHCO03HOCC  
 CCHCCHCO04HCCCCCCHCCHH0CCCCHCCHHCCCCCHCCHH0CCC  
 HCCCHHCCCCCHH0CCCCHCCHHCCCCCHCCHH1CCCCCCHHCCCC  
 CCHH1CCCCCCHHCCCCCHCCHH1CCCCHCCHHCCCCCHCCHH1CCC  
 CCHHCCCCCHCCHH1CCCCHCCHHCCCCCHCCHH1CCCCHCCHHCC  
 CHH1HCCCCCCHCCHH1HCCCCCCHCCHH1HCCCCCCHCCHH2  
 CCCCCCHHCCCCCHCCHH2CCCCCCHHCCCCCHCCHH2CCCCHCCHCO  
 OCCCHCCHH2CCCCHCCHCOOCCCCCHCCHH2CCCCHCCHHCCCCCHC  
 HH2CCCCHCCHHCCCCCHCCHH2HCCCCCCHCCHH2HCCCCCCHC  
 HH2HCCCCCCHCCHH2HCCCCCCHCCHH2HCCCCCCHCCHH2H  
 CCCCCCHCCHH3CCCCCCHHCCCCCHCCHH3CCCCCCHHCCCCCHC  
 HH3CCCCOCHCCHCCHH3CCCCOCHCCHCCHH3CCCCCHCCHC  
 OCCCCCHCCHH3HCCCCCCHCCHH3HCCCCCCHCCHH3HCCCCC  
 CHCCHH3HCCCCCCHCCHH4CCCCOCHCCHCCHH4HCCCCC  
 HCCCHH4HCCCCCCHCCHH4OCCOCCCCCHCCHH4OCCOCCCCCHC  
 H4OCCOCHCCHCCHH4OCCOCHCCHCCHH5HOCCCCCHCCHH5H  
 OCCCHCCHH5OCCOCCCCCHCCHH5OCCOCHCCHCCHH6HOCHC  
 COHCCCCCHCOHCCCCCHCCHCOHCCCCCHCCHCOHCCCCCHC  
 CCC1CCCCCCHHCCCC1CCCCCCHHCCCC1CCCCHCCHHCCCC1  
 CCCCHCCHHCCCC1CCCCHCCHHCCCC2CCCCCCHHCCCC2CCCC  
 CCHHCCCC2CCCCHCCHCOOCHCC2CCCCHCCHCOOCHCC2CCC  
 CCHHCCCC2CCCCHCCHHCCCC2CCCCHCCHHCCCC2CCCCHCCH  
 HCCCC2CCCCHCCHHCCCC2CCCCHCCHHCCCC3CCCCCCHHCCCC  
 3CCCCCCHHCCCC3CCCCCCHHCCCC3CCCCCCHHCCCC3CCCC  
 OOHCCCC3CCCCOCHHCCCC3CCCCHCCHCOOCHCCC3CCCCHCCHC  
 OOHCCC3CCCCHCCHHCCCC3CCCCHCCHHCCCC3CCCCHCCHH  
 CC3CCCCHCCHHCCCC3HCCCCCHCC3HCCCCCHCC3HCCCCCHCC3H  
 CCHCCC3HCCCCCHCC3HCCCCCHCC3HCCCCCHCC3HCCCCCHCC4  
 CCCCCCHHCCCC4CCCCCCHHCCCC4CCCCOCHHCCCC4CCCCO  
 OHCCCC4CCCCHCCHCOOCHCC4CCCCHCCHHCCCC4CCCCHCCH  
 HCCC4HCCCCCHCC4HCCCCCHCC4HCCCCCHCC4HCCCCCHCC4HCC  
 CHCCC4HCCCCCHCC4HCCCCCHCC4HCCCCCHCC4OCCOCHCC4OC  
 COHCCC4OCCOCHHCCC4OCCOCHHCCC5CCCCOCHHCCC5HCCC  
 HCCC5HCCCCCHCC5HCCCCCHCC5HCCCCCHCC5HOCHCCH5HOC  
 HCCC5OCCOCHHCCC5OCCOCHHCCC5OCCOCHHCCC5OCCOCHHCCC6  
 HOCHCCC6HOCHCCC6OCCOCHHCCC6OCCOCHHCCC7HOCHOC0H  
 OCHOC1OCCOCHHOC2CCCCOCHHOC3CCCCHCCHCOOCHOC3OCC  
 OHOC4CCCCCCHHOC4CCCCCCHHOC5CCCCHCCHHOC5CC  
 CHCCHHOC5HCCCCCHOC5HCCCCCHOC6CCCCHCCHHOC6HCCCC  
 OC6HCCCCCHOC7HCCCCOCCO0OCCOCCO1CCCCOCHOC02CC  
 CHCCHCOOCCO2OCCOCHOC03CCCCCCHHOC03CCCCCCH  
 HOC03HOCOCCO4CCCCHCCHHOC04CCCCHCCHHOC04HCC  
 COCCO4HCCCCOCCO5CCCCHCCHHOC05HCCCCOCCO5HCCCCO  
 CO6HCCCCOCCOCH0OCCOCHOC0H1CCCCOCHOC0H1HOC0CC  
 OH2CCCCHCCHCOOCCOCH2OCCOCCOCH3CCCCCCHHOC0H3  
 CCCCCCHHOC0H4CCCCHCCHHOC0H4CCCCHCCHHOC0H4H  
 CCCCCOCH4HCCCCOCCO5CCCCHCCHHOC0H5HCCCCOCCO5  
 HCCCCOCCO6HCCCC

As we can see, the length of the generated string is very large, so we used the *md5* algorithm to generate a *hex digest* for the string. The *hex digest* is 39BF9B334B172E4E71E76B93C830B47E. From this point on, we will only use *hex digest* of *md5* to represent the unique string.

## 4. Results

We implemented the algorithm in the C programming language and tested it on 236,917 chemical compounds in the NCI database of chemical compounds. The input was a trimmed version of a PDB file that contains the atomic symbols and their spatial coordinates. We calculated the bond information and initialized it. We represented the graph in the form of an adjacency matrix and ran our UCK algorithm. The output was then piped to the *md5* algorithm to generate the UCK hex digest.

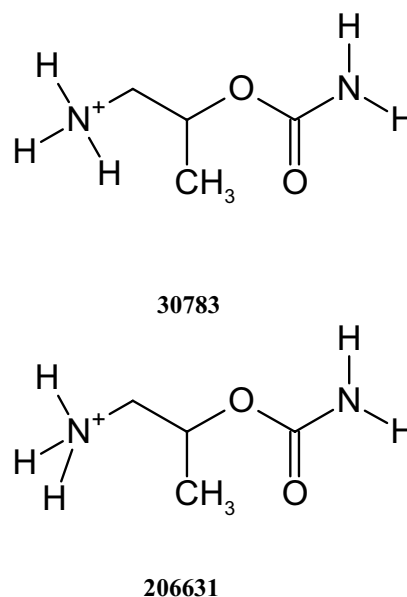
We generated UCK strings and the UCK *md5* hex digest for 236,917 data sets consisting of a variety of drug molecules, including, in many cases, variants of the same molecule. The results are summarized in Table 2. By variants of the same molecule we mean data sets (which represent chemical compounds) that differ in the ordering of the atoms and/or their spatial orientation. The NCI database has many molecules whose structures are very similar but are different molecules. There are also molecules that have the same molecular formula but different structural formulas.

**Table 2.** Summary of number of compounds studied.

<i>Description</i>	<i>Number</i>	<i>Remarks</i>
Total number of chemical compounds.	236,917	Includes some compounds with duplicate entries.
Number of chemical compound with single entry.	203,384	All gave unique UCK.
Number of chemical compound with two or more entries.	33,533	The UCK algorithm gave unique labels to distinct compounds. The UCK algorithm gave the same label to the same compound occurring in multiple entries.

Our algorithm produces excellent results, generating the same key for variants of the same molecule and

different keys for different compounds. The algorithm is invariant to changes in input ordering of atoms and rigid transformations. We were able to recognize molecules that have more than one entry in the database with different database entries. For example, consider two molecules with NCI numbers 30783 and 206631 (Figure 4). Though they have different NCI numbers they are the same molecule. A snapshot of the NCI database is given below, and also the hex digest generated by our algorithm (Table 3). The two data sets have different ordering of atoms and slightly different spatial orientation. Also shown in Table 3 are the hex digest for two molecules with NCI numbers 91771 and 97338 that are the same but different in their respective conformations.

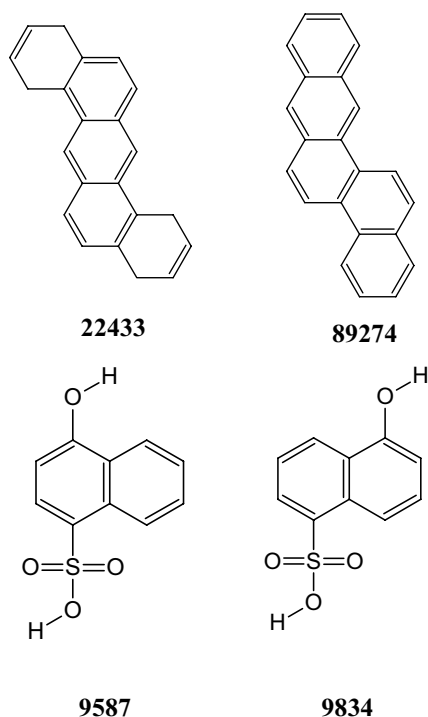


**Figure 4.** Structure of 1-methyl-2-(trimethyl- $\lambda^5$ -aznyl)ethyl carbamate with different NSC numbers.

**Table 3.** Snapshot of NCI database and the hex digest (last column) generated by our algorithm.

<i>NSC Number</i>	<i>Formula</i>	<i>md5 hex digest generation of UCK</i>
30783	C <sub>7</sub> H <sub>17</sub> N <sub>2</sub> O <sub>2</sub>	02994BC7A283073ED9C 1730E2F37EFCD
206631	C <sub>7</sub> H <sub>17</sub> N <sub>2</sub> O <sub>2</sub>	02994BC7A283073ED9C 1730E2F37EFCD
91771	C <sub>38</sub> H <sub>42</sub> N <sub>2</sub> O <sub>6</sub>	5C0F9A8F0ECC0BAF32 CBCA62DA571F42
97338	C <sub>38</sub> H <sub>42</sub> N <sub>2</sub> O <sub>6</sub>	5C0F9A8F0ECC0BAF32 CBCA62DA571F42

The algorithm is sensitive to changes in connectivity even at the remotest portions of the molecule, which makes it very effective in detecting different chemical compounds that are very similar. Figure 5 shows two pairs of compounds that are quite similar. Table 4 gives the results of our algorithm and illustrates the efficiency of the algorithm for molecules with small and difficult to detect changes in structures. We can observe that their UCK keys are different.



**Figure 5.** Structures of dibenzo[a,h]anthracene (22433) and benzo[b]chrysene (89274), 4-hydroxy-1-naphthalenesulfonic acid (9587) and 5-hydroxy-1-naphthalenesulfonic acid (9834).

**Table 4.** Snapshot of NCI database and the hex digest (last column) generated by our algorithm for molecules with changes at remote parts of the molecule.

<i>NSC Number</i>	<i>Formula</i>	<i>md5 hex digest generation of UCK</i>
22433	C <sub>22</sub> H <sub>14</sub>	A15F6359F7AC44C1A90C0 F90598664B4
89274	C <sub>22</sub> H <sub>14</sub>	7A4A4AF922300C12704238 10B748FAF5
9587	C <sub>10</sub> H <sub>8</sub> O <sub>4</sub> S	6BEAF3C856A2C1318F306 DDB0E7F888A
9834	C <sub>10</sub> H <sub>8</sub> O <sub>4</sub> S	64373D3DB8329F56B6AFA E59F07AAFC3

For the largest compound (579 atoms) in the database it took about 5 seconds to generate the unique key. The results for generating unique keys for the largest compound and the 236,917 compounds studied are presented in Tables 5 and 6 respectively. The tests were carried out on a Dell machine with Dual Xeon 2.4 GHz processors with hyper threading, 2 gigabyte of RAM, and approximately 236 gigabyte of disk space on RAID-5 and running Red Hat Linux 8.0 operating system.

**Table 5.** Properties and execution time of the large molecule studied.

<i>Largest Molecule</i>	
NCI Number	57300
# Atoms	579
Formula	C <sub>166</sub> H <sub>328</sub> N <sub>2</sub> O <sub>83</sub>
md5 hex digest	CE6892FDEBE05614AAC08560A5D 4AE8B
generation of UCK	
Time taken for UCK generation	5.204 seconds
Total length of the key (not md5)	5278576 characters

**Table 6.** Total time required to process the entire data set.

<i># Compounds</i>	<i>Time taken (approx.)</i>	<i>Remarks</i>
236,917	18770.16 sec (5.214 hrs)	With other processes taking up to 98.4% CPU and using 'nice'.
236,917	12478.42 sec (3.466 hrs)	With just this job running.

## 5. Summary and Conclusions

We have developed an algorithm called the UCK algorithm that generates unique keys for a wide variety of chemical compounds. The UCK algorithm views molecular structures as undirected labeled graphs. The atoms are represented as the vertices and the edges as the bonds. The algorithm was experimentally tested on 236,917 compounds from the NCI database, and generated unique keys for all the uniquely identified structures. We call these keys Universal Chemical Keys or UCKs. We have used the UCKs to build distributed web-service based applications involving protein docking

with data pulled from multiple distributed databases. Without a unique key such as the UCK, the application would have no way of knowing whether two distributed proteins or chemicals were the same or not.

The UCK algorithm depends only upon the structure of the labeled graph. Distinct labeled graphs give rise to distinct UCKs. On the other hand, two different labeled graphs could, in theory, give rise to the same UCK. This is a calculated trade-off. Distinguishing arbitrary labeled graphs is NP-hard and hence a fast deterministic algorithm cannot be expected. On the other hand, chemical compounds give rise to a restricted class of labeled graphs - it is likely that our algorithm can be proven to be unique on various restricted classes of chemical compounds. We plan to investigate these types of results and to verify experimentally the properties of our UCK algorithm on additional databases of chemical compounds.

In particular, in this study the UCK algorithm uses a depth  $d = 2$ , for labeling the vertices. Using different depths and dynamically assigning the depth depending upon the compound gives rise to UCKs that are more expensive to compute but stronger. We plan on investigating these trade-offs in future work.

To summarize, the UCK algorithm is a fast and effective algorithm that provides intrinsic and unique keys for a wide class of commonly occurring chemical compounds.

## 6. References

- [1] R. Rivest. *The MD5 message digest algorithm*, RFC1321, 1992.
- [2] IUPAC, *Nomenclature of Organic Chemistry*, Pergamon Press: Oxford, 1979.
- [3] M. H. Klin, O. V. Lebedev, T. S. Pivina and N. S. Zefirov, "Nonisomorphic cycles of maximum length in a series of chemical graphs and the problem of application of IUPAC nomenclature rules", *MATCH* 27, 1992, pp. 133-151.
- [4] R. C. Read, "The Coding of various Kinds of unlabeled Trees", *Graph Theory and Computing*, Academic Press, New York, 1972.
- [5] R. C. Read, "A new system for the design of chemical compounds. Coding of cyclic compounds", *Journal of Chem. Inf. and Comp. Sci.*, 25, 1985, pp. 116-128.
- [6] M. Randić, "On recognition of identical graphs representing molecular topology", *J. Chem. Phys.*, 60, 1974, pp. 3920-3928.
- [7] A. Prokurowski, "Search for unique incidence matrix of a graph", *BIT*, 14, 1974, pp. 209-226.
- [8] *Chemistry International*, 23, 3, 2001.
- [9] V. L. Arlazarov, I. I. Zuev, A. V. Uskov and I. A. Faradzev, "An algorithm for the reduction of finite non-oriented graphs to canonical form", *Zh. Vychisl. Mat. Mat. Fiz.*, 14, 3, 1974, pp. 737-743.
- [10] T. Beyer and A. Proskurowski, "Symmetries in graph coding problem", *Proc. NW76 ACM/CIPC Pac. Symp.*, 1976, pp. 198-203.
- [11] C. Böhm and A. Santolini, "A quasi-decision algorithm for the p-equivalence of two matrices", *ICC BULLETIN*, 3, 1 1964, pp. 57-69.
- [12] M. Klin, C. Rucker, G. Rucker and G. Tinhofer, "Algebraic combinatorics in mathematical chemistry. Methods and algorithms. I. Permutation Graphs and coherent (Cellular) algebras", *Technical Report, Technische Universität München, TUM-M9510*, 1995.
- [13] B. D. McKay, "Computing automorphism and canonical labeling of graphs", *International Conference on Combinatorial Mathematics*, Canberra (1977), Lecture Notes in Mathematics 686, Springer-Verlag pp. 223-232.
- [14] B. D. McKay, "Practical Graph Isomorphism", *Congressus Numerantium*, 30, 1981, pp. 45-87.

---

† Robert Grossman is also with Open Data Partners.